Event Detection Services Using Data Service Middleware in Distributed Sensor Networks

Shuoqi Li, Sang H. Son, and John A. Stankovic

Department of Computer Science, University of Virginia, USA
{sl7q, son, stankovic}@cs.virginia.edu

Abstract. This paper presents the Real-Time Event Detection Services middleware which is a component of the Data Service Middleware (DSWare). DSWare is a data-centric and group-based service for sensor networks. The real-time event service includes unreliability of individual sensor reports, correlation among different sensor observations, and inherent real-time characteristics of events. The event service supports confidence functions which are designed based on data semantics, including relative importance of sub-events and historical patterns. When the failure rate is high, the event service enables partial detection of critical events to be reported in a timely manner. It can also be applied to differentiate between the occurrences of events and false alarms.

1 Introduction

Sensor networks are large-scale wireless networks that consist of numerous sensor and actuator nodes used to monitor and interact with physical environments [11][14]. From one perspective sensor networks are similar to distributed database systems. They store environmental data on distributed nodes and respond to aperiodic and long-lived periodic queries [7][15][20]. Data interest can be pre-registered to the sensor network so that the corresponding data is collected and transmitted only when needed. These specified interests are similar to views in traditional databases because they filter the data according to the application's data semantics and shield the overwhelming volume of raw data from applications [8][26].

Sensor networks also have inherent real-time properties. The environment that sensor networks interact with is usually dynamic and volatile. The sensor data usually has an absolute validity interval of time after which the data values may not be consistent with the real environment. Transmitting and processing "stale" data wastes communication resources and can result in wrong decisions based on the reported out-of-date data. Besides data freshness, often the data must also be sent to the destination by a deadline. To date, not much research has been performed on real-time data services in sensor networks.

Despite their similarity to conventional distributed real-time databases, sensor networks differ in the following important ways. First, individual sensors are small in size and have limited computing resources. They also must operate for long periods of time in an unattended fashion. This makes power conservation an important concern in prolonging the lifetime of the system. In current sensor networks, the major source of power consumption is communication. To reduce unnecessary data transmission from each node, data collection and transmission in sensor networks are always initiated by subscriptions or queries. Second, any individual sensor is not reliable. Sensors can be damaged or die after consuming the battery. The wireless communication medium is also unreliable. Packets can collide or be lost. Because of these issues we must build trust on a group of sensor nodes instead of any single node. Previous research emphasizes reliable transmission of important data or control packets at the lower levels, but less emphasis is on the reliability on data semantics at the higher level [23]. Third, the large amount of sensed data produced in sensor networks necessitates in-network processing. If all raw data is sent to base stations for further processing, the volume and burstiness of the traffic may cause many collisions and contribute to significant power loss. To minimize unnecessary data transmission, intermediate nodes or nearby nodes work together to filter and aggregate data before the data arrives at the destination. Fourth, sensor networks can interact with the environment by both sensing and actuating. When certain conditions are met, actuators can initiate an action on the environment. Since such actions are difficult to undo, reducing false alarms is crucial in certain applications.

The remainder of this paper is organized as follows: In section 2, we present related work. In section 3, we present the design of Data Service Middleware (DSWare) and some major components of DSWare. DSWare is a specialized layer that integrates various real-time data services for sensor networks and provides a database-like abstraction to applications. In section 4 we then present a detailed description of the event detection mechanism. Event detection is one of the most important data services in sensor networks because it is a way to "dig" meaningful information out of the huge volume of data produced. It aims to find the "right data" at the "right place" and ensure the data is sent at the "right time". Event Detection Services in DSWare associate a confidence value with each decision it makes based on a pre-specified confidence function. It incorporates the unreliability of sensor behavior, the correlation among different factors, and reduces false alarms by utilizing data semantics. Section 5 presents the preliminary evaluation of the event detection mechanism. We conclude the paper in Section 6.

2 Related Work

There are many ongoing middleware research projects in the area of sensor networks, such as Cougar, Rutgers Dataman, SINA, SCADDS, Smart-msgs, and some virtual-machine-like designs [1][2][3][4][8][12][17][26]. COUGER and SINA are two typical data-centric middleware designs which have goals that are similar to our design goal of providing data services. In COUGER, sensor data is viewed as tables and query execution plans are developed and possibly optimized in this middleware. Our work on DSWare is more tailored to sensor networks, including supporting group-based decision, reliable data-centric storage, and implementing other approaches to improve the performance of real-time execution, reliability of aggregated results and reduction of communication. SINA is a cluster-based middleware design which focuses on the cooperation among sensors to conduct a task. Its extensive SQL-like primitives can be used to issue queries in sensor networks. However, it does not provide schemes to hide the faulty nature of both sensor operations and wireless communication. In SINA it is the application layer that must provide robustness and reliability for data services. In DSWare, the real-time scheduling component and built-in real-time features of other service components make DSWare more suitable than SINA for real-time applications in ad hoc wireless sensor networks.

Multisensor data fusion research focuses on solutions that fuse data from multiple sensors to provide more accurate estimation of the environment [16][22]. In mobile-agent-based data fusion approaches, software that aggregates sensor information are packed and dispatched as *mobile agents* to "hot" areas (e.g., the area where an event occurred) and work independently there. The software migrates among sensors in a cluster, collects observations, then infers the real situation [22]. This approach and our group-based approach both make use of consensus among a number of nearby sensors of the same type to increase the reliability of a single observation. The mobile-agent-based approach, however, leverages on the migration traffic of mobile agents and their appropriate processing at each sensor node in its routes. For instance, if a node in the route inserts wrong data or refuses to forward the mobile agents, the aggregation and subsequent analysis are untrustful. Our approach does not have such limitations: malfunctioning of individual nodes does not infect the entire group.

A fuzzy modelling approach is sometimes used for data fusion in sensor networks. It is used to model the uncertainty in sensor failures and faulty observations [25]. This approach is useful in modelling the sensor error rates due to equipment wear and aggregating local decisions from multiple sensors that measure the same type of data. Some optimal decision schemes focus on the fusion of asynchronously arriving decisions [10][24]. E. Bosse et. al. presented a modelling and simulation approach for a real-time algorithm in multi-source data fusion systems [9]. These data fusion schemes are suitable for increasing the accuracy of decisions, but require extensive computing resources. In our approach to event detection, the computation in fusion nodes is small.

Dempster-Shafer evidential theory is also applied to incorporate uncertainty into decisions in some sensor fusion research [21]. This scheme uses *Belief* and *Plausibility* functions to describe the reliability feature of each source and uses a normalized Dempster's combination rule to integrate decisions from different sources. Our *confidence function* is similar to Dempster-Shafer method except that we place the evidence in both temporal and spatial spectrums to take data real-time validity intervals and possible contexts into consideration.

3 Data Service Middleware (DSWare)

A data services middleware can avoid re-implementing the common data service part of various applications. We develop a Data Service Middleware (DSWare) Layer that exists between the application layer and the network layer. This middleware provides data service abstraction to applications, as depicted in Fig. 1. In this architecture, routing is separated from both DSWare and the network layer since the group management and scheduling components in DSWare can be used to enhance the power-awareness and real-time-awareness of routing protocols. Fig. 2 demonstrates the architecture of DSWare.

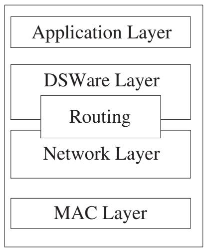

Fig. 1. Software Architecture in Sensor Networks

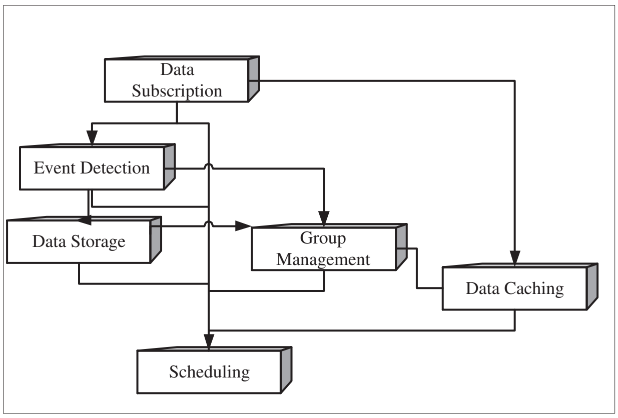

Fig. 2. Framework of DSWare

3.1 Data Storage

Data-centric storage is an implementation of a data storage service [23]. Data that describes different occurrences of some type of activity can be mapped to certain locations so that future queries for this type of data do not require being flooded to the whole network. The Data Storage Component in DSWare provides similar mechanisms to store information according to its semantics with efficient data lookup and supports more robustness during node failures. Correlated data can be stored in geographically adjacent regions to enable possible aggregation and in-network processing.

- Data Lookup
  We use two levels of hashing functions to map data to physical storage nodes. Each type of data has its unique identifier (e.g, the activity name string and the object's privilege profile) and it is used as key for the first level hashing function. The first level hash function maps the key to a logical storage node in the overlay network. At this level, storage nodes establish a hierarchy. In DSWare, we have one more hashing procedure to map a single logical node to multiple physical nodes. When a base station sends queries for this data, the information is fetched from one of these physical locations. Future designs need to consider how to map related data to geographically adjacent locations to promote data aggregation and in-network processing.
- Robustness
  Data stored in an individual node can be lost due to disaster, node damages, energy shortage, and other reasons. If we map a certain type of data to an individual node, when this activity occurs, lots of event data is sent to this node during a short period. The burst of traffic will lead to high collision and power consumption in the storage vicinity and indirectly decrease the reliability and availability of the storage node. In DSWare, data is replicated in multiple physical nodes that can be mapped to a single logical node. Queries are directed to any of these nodes to avoid high traffic collision and heavy load pushed on a single storage node. Load is balanced among the set of physical nodes and the lifetime of an individual node is prolonged. The consistency among these nodes is a key issue for a data storage component. To avoid peak time traffic, we choose "weak consistency" among the nodes. Most of the data on these nodes are identical except a small portion of the newest data. This new data is eventually propagated to the other peer nodes. The size of the portion of data that is inconsistent is bounded and nodes do the replication when their own work load is low.

3.2 Data caching

The Data Caching Service provides multiple copies of the data most requested. This data is spread out over the routing path to reduce communication, increase availability and accelerate query execution [5]. It uses a simplified feedback control scheme to dynamically decide whether to place copies of the data around the frequently queried nodes.

There is a tradeoff between the query response time and maintenance overhead of data copies. A node can use the total number of queries routed through itself, the proportion of periodic queries, average response time from the data source, the number of copies that already exist in the neighborhood and other observations as inputs to the controller at a node and the controller determines whether to keep a copy. The data caching service in DSWare monitors current usages of the copies and determines whether to increase or reduce the number of copies and whether to move some copies to another location by exchanging information in the neighborhood.

3.3 Group Management

The Group Management component provides localized cooperation among sensor nodes to accomplish a more global objective. There are several reasons why group management is important. First, normally functioning sensors within a geographic area provide similar sensor values. A value that most nodes in a group agree on should have higher confidence, than a value that is in dispute or varies widely. Second, based on the similar observations by nearby sensors in a sufficiently dense area, we can recognize the nodes that keep reporting erroneous results. We may discard the suspicious nodes in later coordination and computations to provide more reliable measurement. Third, some tasks require cooperation of multiple sensors. Movement and speed approximations require more than one sensor to combine their observations to calculate the direction and velocity. Finally, when a region has adequate density of sensors, a portion of them can be put into sleep mode to save energy.

Based on the different reasons discussed above, there are different ways to formulate a group. For most tasks, groups are formed as the query is sent out and dissolved when the query is expired or the task is accomplished. In this case, the group formulation criterion is sent to the queried area first. Nodes decide whether to join this group by checking whether they match the criterion. Some groups are relatively stable after formulation, such as those measuring temperature. Some groups are more dynamic, such as the groups tracking the movement of a vehicle [6]. For a dynamic group, changed criterion is broadcast persistently in a small area whose center is the current group. Hence, nodes can join and leave the group when the target moves. There are other groups designed for geographically stable goals. These groups are not sensitive to tasks, so they can be formulated during system deployment or when explicitly specified by the applications. These groups are not necessary for the accomplishment of a task, but they have significant effects in reliability and reduction of energy consumption and communication.

3.4 Event Detection

In the event detection service, events are pre-registered according to the specific application. Event detection is a common and important service in sensor networks. We present a detailed protocol for event detection in section 4.

3.5 Data Subscription

As a type of data dissemination service, Data Subscription queries are very common in sensor networks. These queries have their own characteristics, including relatively fixed data feeding paths, stable traffic loads for nodes on the paths, and possible merges of multiple data feeding paths. For example, a base station embedded in a policeman's PDA sends a subscription request to the sensor network : "Show me the traffic status at the crossing of Ivy Road and Alderman Road and keep providing the traffic information every 3 minutes for the next two hours (query duration)." In this case, the base station subscribes to the data of node A for duration D (two hours) and rate R (1 per 3 minutes). When several base stations subscribe for the data from the same node at different rates, the Data Subscription Service places copies of the data at some intermediate nodes to minimize the total amount of communication. It changes the data feeding paths when necessary, as shown in Fig. 3.

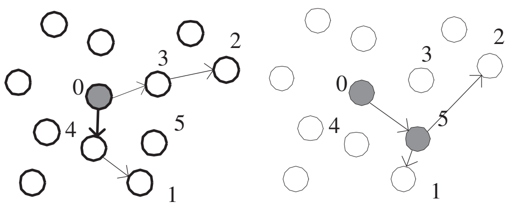

Fig. 3. When there are multiple subscribers (node 1 and node 2) for the data at node 0, the Data Subscription Service detects the proximity of the two paths and merges these two paths by placing a copy of the data at node 5 and lets node 5 send data to the two subscribers during each requesting interval.

3.6 Scheduling

The Scheduling component is a special component because it provides the scheduling service for all components in DSWare. Two most important scheduling options are energy-aware and real-time scheduling. By default, we apply a real-time scheduling mechanism (EDF, EDDF, with or without admission control) as the main scheduling scheme because most queries in sensor networks are inherently real-time tasks. We can also apply the energy-aware mechanism when we have already met the requirements of real-time scheduling. Applications can specify the actual scheduling schema in the sensor networks based on the most important concerns.

4 Event Detection Services

In this section, we present the event detection services of DSWare. We first discuss some of the key concepts of the event detection services, including event hierarchy, confidence, and time semantics, followed by implementation issues.

4.1 Event Hierarchy

An *observation* is the low-level output of a sensing device during a sensing interval. It is a measurement of the environment. An event is an activity that can be monitored or detected in the environment and is of interest to the application.

We group events into two different types: *atomic events* and *compound events*. An *atomic event* refers to an event that can be determined merely based on an observation of a sensor. Suppose we have registered the following events:

High temperature event represents the observation that the temperature is higher than a specified threshold.
Light event represents an occurrence of a sharp change in the light intensity.
Acoustic event represents the occurrence of an unusual sound matching a certain signature.
Explosion event might be defined as the three events above are reported in the same region within a specified time interval.

In this example, whether a high temperature event occurs or not can be determined from an observation of a single temperature sensor. Such event is an atomic event.

A *compound event* can not be detected directly from observations; instead, it must be inferred from detections of other atomic or compound events(i.e. sub-events of this compound event). In the example above, the explosion event is a compound event. High temperature, light and acoustic events are sub-events of the explosion event.

4.2 Confidence, Confidence Function and Phase

When a compound event occurs, it is possible that not all sub-events are detected. For example, when an explosion actually occurs, only two atomic sub-events – the high temperature and the light sub-events – could be detected, if the sensors that detect the acoustic signals are damaged in the explosion. We use the notion of *confidence* to address this problem.

A *confidence function* takes whether the sub-events have been reported or not as boolean parameters and produces a numeric value of output based on the event's semantics. The *confidence* is the return value of the confidence function specified in event registration. An event with a confidence higher than 1.0 is regarded as "confirmed", i.e., the sensor network is highly confident that the event actually occurred.

A *confidence function* specifies the relationships among sub-events of a compound event with other factors that affect the decision such as relative importance, sensing reliability, historic data, statistical model, fitness of a known pattern and proximity of detections. The information is derived from event semantics in real life. A confidence function can be a simple linear equation or a complex statistical model. For example, if the temperature has been continuously going up for a period of time, combined with light sub-event, then a report of fire event carries a higher confidence compared to the report that is based on the observations only on temperatures going up and down rapidly in a short period of time.

In reality, an event always has its meaningful contexts, which can be modelled using a Finite State Machine (FSM). For example, in a residential monitoring system, morning, afternoon, and evening can be the states of this system. We call these states *phases*. In each *phase*, there is a set of events that are likely to occur with meaningful context, while other events are less likely to occur [27]. Consider a chemical factory. Dissemination of a chemical might not happen except during a specific production phase. If all sub-events of this chemical event are detected during a phase in which the event is very unlikely to happen, the system could either give this event detection a low confidence or report the possible malfunction of the sensors. Using phases in this manner not only saves power in monitoring and event detection, but also increases the reliability of event detection.

4.3 Real-Time Semantics

Each sub-event has an *absolute validity interval (avi)* associated with it. The avi depicts the temporal consistency between the environment and its observed measurement. Continuing the explosion example, the temperature sub-event can have a longer avi because high temperature usually will last for a while, while the light sub-event may not last long because in an explosion, a sharp increase in the intensity of light would happen only for a short period of time. It is the responsibility of the application developer to determine the appropriate avi values.

When an event consists of more than one sub-event, the time an aggregating node should wait for the arrivals of all these sub-events becomes an important issue. The delay of a sub-event's detection varies according to sensors' sampling period and communication delay. We should preserve a time window to allow all possible reports of sub-events to arrive to the aggregating node. Wireless media and unpredictable environment in which a sensor network exists make both the loss of messages and failures of nodes common. For this reason, we can't risk reporting an urgent event late. If before the timer expires the confidence value has reached 1, the event is reported to registrants without waiting any more. If the confidence value exceeds the min_confidence value specified in sub-event list when the timer expires, the event is reported to registrants with this confidence value. If the confidence value hasn't reached the min_confidence value when the timer expires, the event is not reported.

After an event is detected, it should be sent to the registrants before the reporting deadline. For example, we can use the Velocity Monotonic Scheduling or SPEED protocol [13][18].

4.4 Registration and Cancellation

To register an event of interest, an application submits a request in the following SQL-like statement:

```
INSERT INTO EVENT_LIST
     (EVENT_ID, RANGE_TYPE, DETECTING_RANGE,
       SUBEVENT_SET, REGISTRANT_SET, REPORT_DEADLINE,
       DETECTION_DURATION [, SPATIAL_RESOLUTION ]
        [, ACTIONS])
VALUES     ()
```

Range_Type and Detecting_Range together specify a set of sensor nodes that should be responsible for detecting this event. The Range_Type can be GROUP or AREA. The Detecting_Range is the group's description (e.g., Group ID) or the area's coordinates' range. If an application specifies an area in its registration request, one or more groups will be established in this area. Because of the limited space, we cannot describe different options of group formulations and their contexts in this paper. It will be covered in a separate paper. When an event is detected, it will be reported before the Report_Deadline to every node in the Registrant_Set. If an application receives an event detection report with a expired Report_Deadline, it can decide whether to ignore this "stale" report, or take it and reduce its associated confidence. Detection_Duration denotes the ending time for this event detection task. After the duration time, the event's information is void and nodes stop detecting this event. Event information will be deleted from this group or area. Temporary groups built for this event are dissolved. The Spatial_Resolution defines the geographical granularity for the event's detection. The Subevent_Set defines a set of sub-events and their timing constraints. Here we give its definition:

```
Subevent_Set { Time_window,
               Phase_set,
               Confidence_function,
               Min_confidence,
               (sub-event_1, avi1),
               [(sub-event_2, avi2),...]}
```

The Time_window specifies the time interval during which the sub-events reports are collected. The Phase_set identifies the phase to which the event belongs. The Confidence_function and Min_confidence represent the function to be used for computing the confidence and the minimum confidence required to report the occurrence of the sub-event, respectively.

Let P denote the current Phase in the group or area and S denote the set of sub-events for event E, i.e., $S = (sub - event1, sub - event2...)$.

E is detected when the following are true:

1) P belongs to Phase_set of E.

2) For every s in S, calculate B(s): B(s) = 1 when s has been detected and (current_time - detected_time) $\leq$ avi of s; B(s) = 0 otherwise.

3) Calculate confidence = f(B($s1$), B($s2$), ...,), where f is the confidence function.

4) When Time_window expires: if (confidence $\geq$ min_confidence) report the event with confidence value.

Registered events can be cancelled even before the Detection_duration is terminated by submitting a cancellation request. Event cancellation is similar to event detection. The difference is that it only needs to specify the event's id instead of describing an event's criteria.

```
DELETE FROM EVENT_LIST
    WHERE EVENT_ID = event_id
```

After an event is cancelled, the event's information is void and nodes stop detecting the event. Event information is deleted from the group or area. Any groups assembled for this event are dissolved.

4.5 Discussion

In the current version of the prototype, we made some simplifications to demonstrate the main ideas on data semantics, real-time constraints, and reliability of decisions. We understand the complexity and various choices on issues including the formats for registration and cancellation, group formations, confidence function, and spatial/temporal resolutions. In this part, we provide some discussion on important issues in event detection services.

SQL-like Language in Event Detection: As presented in Section 4.4, we use SQL-like statements for the registration and cancellation of an event. This approach provides a simple interface for applications [8][19]. The syntax of the statements is the same as standard SQL statements. So the application can insert events to a traditional database or a sensor network without any changes in the code. This is effective for applications that need event detection services, without paying any special attention to the actual type of the database and data service middleware that is providing the service.

In some cases, this approach is unsuitable because of its parsing overhead. After an SQL-like statement is issued, DSWare parses it, generates an execution plan, and calls the corresponding methods to execute the registration, execution, and cancellation. Parsing consumes memory and processing power. For sensor networks in which sensors are very limited in processing and memory capacities, it might be better to provide method signatures to applications instead of standard SQL. However, we believe that the SQL-like approach is the right one, since it provides the flexibility and expressiveness of SQL to cover a large number of possible event specifications. This is the main reason why we include an optional SQL-parser module in our DSWare.

Spatial and Temporal Resolutions: Spatial resolution indicates the possible detection radius of an event. If the size of a detection group is too small compared to this event, there might be several groups in this event's coverage that report this event. The Event Detection component should be able to tell whether these are different occurrences or just repeated reports of a same event.

Temporal resolution has a similar property to spatial resolution, except that it specifies the detection granularity in the time dimension. Some events last much longer than the sensing interval of a sensor. It is unnecessary for some applications to report a single occurrence repetitively. For example, an application sets the temporal resolution of a fire event as 10 minutes. At the beginning of the fire, the group that detects the fire reports the fire event to the registrants. Assume that there is some mechanism to guarantee that the registrants have received the report, this group can ignore any subsequent occurrence of this event's sub-events within 10 minutes, because that is possibly the same event. The temporal resolution is not required for every application because some applications require the sensors to report an event's existence no matter whether it is a new one or not.

5 Evaluation of Real-time Event Detection Services

For the preliminary evaluation, we have implemented the real-time event detection services in GloMoSim[29]. Within a terrain of $2000 * 2000m^2$, which is uniformly divided into 16 groups, we place 100 sensor nodes to sense temperature, light or acoustics. The simulator simulates the detection of an Explosion(E) event that consists of a high temperature atomic event(T), a light atomic event(L) and an acoustic atomic event(A). T and A are modelled as circles whose coverage radius expands over time, denoting the actual energy expansion in a real system [28]. L is modelled as spatially distributed events that occur repetitively during explosions with a very short lifetime.

To simulate the error distribution around a hazard event as an explosion, the failure rate of sensors decreases quadratically with the distance between a sensor and the center of explosion. The Explosion event is registered by node 1 (at upper-left corner of the terrain) to the entire network. In our simulation, we assume high temperature is a more consistent indicator of an explosion among the three sub-events and temperature sensing devices are more robust in the physical environment. Accordingly, we specify a simplistic confidence function as follows:

$$Confidence_E := 0.6 * B(T) + 0.3 * B(L) + 0.3 * B(A). \tag{1}$$

B(x)=1 if x is detected within time window of 3 seconds; 0 otherwise.

The weights of sub-events are consistent with our application and experimental settings. The min_confidence is set as 0.9, which means an explosion event will be reported if the $Confidence\_E$ is not less than 0.9.

5.1 Performance in Reduction of Communication

Previous work [11] uses group coordination or in-network processing to reduce the number of reports generated from a group. In the best case, only one report of a environment property is generated from a group during each sensing interval. However, for correlated events, such an algorithm has to detect the events separately, send all reports (at least one for each atomic event) to an outside node and the entire analysis will be done there. We use this algorithm as our Baseline in the experiments. Fig 4 is the comparison between real-time event services (denoted by the DSWare curve in the figure) and the Baseline on the number of messages transmitted in the network and the explosion reports received by the registrant. In this experiment, explosion events appear randomly with an initial radius of 300m. The figure demonstrates that an event detection scheme which is established upon data and application semantics can further process and aggregate data and thus reducing unnecessary communication without sacrificing real-time constraints.

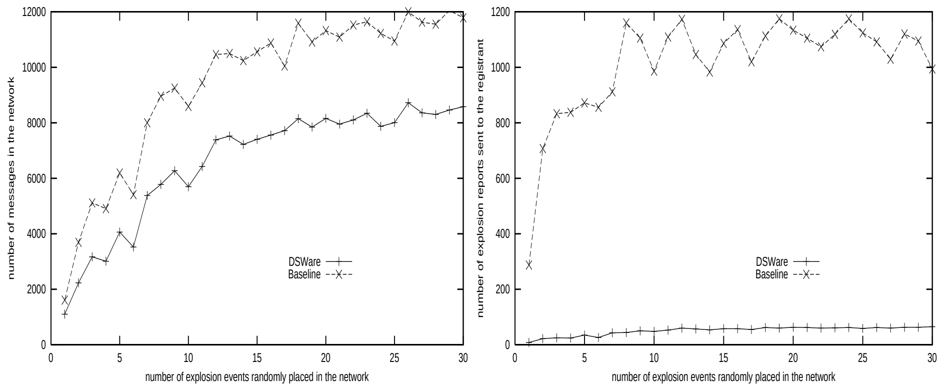

Fig. 4. Comparison between DSWare and Baseline in Communication

5.2 Performance in Differentiating Events and Event-like Factors

One of the key features of the real-time event services is that it can be used to differentiate between events and pseudo events (event-like factors [28]) happening in physical environment to avoid false alarms.

In this experiment, we place an explosion event with initial radius of 300m at (1000.0, 1000.0) in each run. We place High Temperature events with initial radius of 150m randomly in the network and plot the number of reports sent to the registrant in Fig 5. The figure indicates that with time windows and a carefully designed confidence function, our approach can achieve a performance close to ideal detection (no false alarms). The slight difference between the DSWare reports and the ideal case reports is due to the randomness of sensor types in the explosion event area.

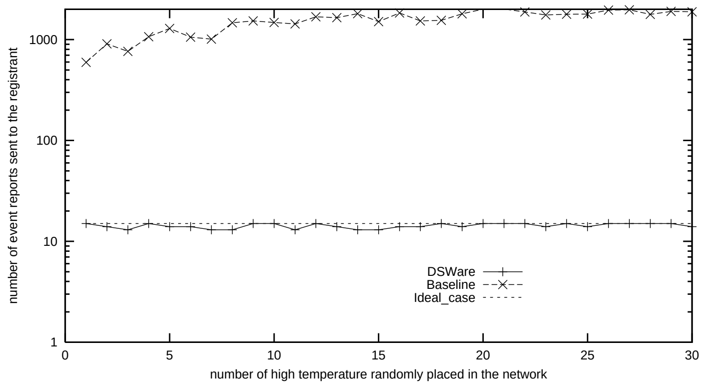

Fig. 5. Performance in Differentiating Events and Event-like Factors

6 Conclusions

A sensor network should be able to provide the abstraction of data services to applications. However, because of the lack of basic data-centric services in sensor networks, current applications need to implement the entire stack of application-specific data services including group management, query optimization, local data processing, and event detection. Such a tight coupling of data services and application logic has several disadvantages and increases the complexity of applying sensor networks as databases in a large software system. We have developed a data-centric service middleware in sensor networks called DSWare. DSWare is a flexible middleware designed to hide unattractive characteristics of sensor networks including the unreliability of individual sensing and communication, complexity and necessity of group coordination, and large volume of dynamic data distributed all over the networks, to present a more general data service interface to applications. Applications are freed from complicated low level operations of sensor networks and are able to retrieve data from sensor networks using similar interfaces as conventional databases.

Event detection is a one of the services that is most widely used in sensor network applications. Instead of providing only simple detection of atomic events, we have developed a middleware architecture that accommodates the data semantics of real-life compound events and tolerates the uncertainty and faultiness in sensor networks.

The current version of DSWare including the event detection services is the first step to deliver a flexible and efficient data service middleware for sensor networks. Our future work includes extending the event detection services to support applications for mobile event tracking and implementing other services in DSWare.

References

1. Cougar Project. www.cs.cornell.edu/database/cougar.
2. Rutgers Dataman Project. www.cs.rutgers.edu/dataman.
3. SCADDS: Scalable Coordination Architectures for Deeply Distributed Systems. www.isi.edu/scadds.
4. Smart Messages Project. discolab.rutgers.edu/sm.
5. S. Bhattacharya, H. Kim, S. Prabh, and T. Abdelzaher. Energy-Conserving Data Placement and Asynchronous Multicast in Wireless Sensor Networks. In *Proceedings of the 1st International Conference on Mobile Systems, Applications, and Services*, San Francisco, CA, 2003.
6. B. Blum, P. Nagaraddi, A. Wood, T. Abdelzaher, S. H. Son, and J. A. Stankovic. An Entity Maintenance and Connection Service for Sensor Networks. In *Proceedings of the 1st International Conference on Mobile Systems, Applications, and Services*, San Francisco, CA, 2003.
7. P. Bonnet, J. Gehrke, and P. Seshadri. Querying the Physical World. *IEEE Personal Communication Magazine*, (7):10–15, Oct 2000.
8. P. Bonnet, J. Gehrke, and P. Seshadri. Towards Sensor Database Systems. In *Proceedings of the 2nd International Conference on Mobile Data Management*, Hong Kong, 2001.
9. E. Bosse, J. Roy, and S. Paradis. Modelling and Simulation in Support of Design of a Data Fusion System. *Information Fusion*, (1):77–87, Dec 2000.
10. W. Chang and M. Kam. Asynchronous Distributed Detection. *IEEE Transactions on Aerospace Electronic Systems*, pages 818–826, 1994.
11. D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. In *Proceedings of the 5th Annual International Conference on Mobile Computing and Networks*, Seattle, WA, 1999.
12. J. Feng, F. Koushanfar, and M. Potkonjak. System-Architectures for Sensor Networks: Issues, Alternatives, and Directions. In *Proceedings of the 20th International Conference on Computer Design*, Freiburg, Germany, 2002.
13. T. He, J. A. Stankovic, C. Lu, and T. Abdelzaher. SPEED: A Stateless Protocol for Real-Time Communication in Ad Hoc Sensor Networks. In *Proceedings of the 23rd International Conference on Distributed Computing Systems*, Providence, RI, 2003.
14. J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System Architecture Directions for Networked Sensors. *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.
15. C. Jaikaeo, C. Srisathapornphat, and C-C Shen. Querying and Tasking in Sensor Networks. In *Proceedings of SPIE's 14th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Control (Digitization of the Battlespace V)*, Orlando, FL, 2000.

16. D. Jayasimha, S. Ivengar, and R. Kashyap Information Integration and Synchronization in Distributed Sensor Networks. *IEEE Transactions on Systems, Man and Cybernetics*, 21(5):1032–1043, Sep/Oct 1991.
17. F. Mattern, K. Rőmer, O. Kasten. Middleware Challenges for Wireless Sensor Networks. *ACM SIGMOBILE Mobile Computing and Communication Review (MC2R)*, 2002.
18. C. Lu, B. Blum, T. Abdelzaher, J. A. Stankovic, and T. He. RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks. In *Proceedings of the 8th IEEE Real-Time Technology and Applications Symposium*, San Jose, CA, 2002.
19. S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks. In *Proceedings of the 18th International Conference on Data Engeneering*, San Jose, CA, 2002.
20. S. Madden and M. J. Franklin. Fjording The Stream: An Architecture for Queries over Streaming Sensor Data. In *Proceedings of the 18th International Conference on Data Engeneering*, San Jose, CA, 2002.
21. P. R. Murphy. Dempster-Shafer Theory for Sensor Fusion in Autonomous Mobile Robots. *IEEE Transactions on Robotics and Automation*, 14(2):197–206, Apr 1999.
22. H. Qi, X. Wang, S. S. Iyengar, and K. Chakrabarty. Multisensor Data Fusion in Distributed Sensor Networks Using Mobile Agents. In *Proceedings of 5th International Conference on Information Fusion*, Annapolis, MD, 2001.
23. S. Ratnasamy, D. Estrin, R. Govindan, B. Karp, S. Shenker, L. Yin, and F. Yu. Data-Centric Storage in Sensornets. In *Proceedings of the 1st Workshop on Sensor Networks and Applications*, Atlantic, GA, 2002.
24. V. N. S. Samarasooriya and P. K. Varshney. A Sequential Approach to Asynchronous Decision Fusion. *Optical Engineering*, 35(3):625–633, Mar 1996.
25. V. N. S. Samarasooriya and P. K. Varshney. A Fuzzy Modeling Approach to Decision Fusion under Uncertainty. *Fuzzy Sets and Systems*, 114(1):59–69, Aug 2000.
26. C-C Shen, C. Srisathapornphat, and C. Jaikaeo. Sensor Information Networking Architecture and Applications. *IEEE Personel Communication Magazine*, 8(4):52–59, Aug 2001.
27. M. R. Tremblay and M. R. Cutkosky. Using Sensor Fusion and Contextual Information to Perform Event Detection during a Phase-Based Manipulation Task. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Pittsburgh, PA, 1995.
28. S. Yan, S. Wang, and Z. Dou. An Energy Model in Fire Detection and Integrated Analysis on False Alarms. In *Proceedings of the 12th International Conference on Automatic Fire Detection*, Gaithersburg, MD, 2001.
29. X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulation*, Alberta, Canada, 1998.