Ubiquitous Computing: Challenges in Flexible Data Aggregation

Eiko Yoneki and Jean Bacon

University of Cambridge Computer Laboratory
Cambridge CB3 0FD, United Kingdom
{Eiko.Yoneki, Jean.Bacon}@cl.cam.ac.uk

Abstract. A dramatic increase of event monitoring capabilities by wireless sensors requires new, more sophisticated, event correlation over time and space. This new paradigm implies composition of events in ubiquitous computing environments and event Correlation will be a multi-step operation from event sources to final subscribers, combining information collected by wireless devices into higher level information or knowledge. We define generic composite event semantics, which extend traditional event composition with data aggregation in wireless sensor networks (WSNs). This work bridges data aggregation in WSNs with event correlation services over distributed systems. We use interval-based semantics for event detection, defining precisely complex timing constraints.

1 Introduction

In event-based middleware systems, an event correlation service allows consumers to subscribe to patterns of events (composite events). This provides an additional dimension of data management, and improvement of scalability and performance in distributed systems. Particularly in wireless networks, providing event correlation as a middleware service helps to simplify the application logic and reduce its complexity. Event correlation is also important for constructing reactive distributed applications.

The recent evolution of ubiquitous computing has brought with it a significant increase of event monitoring capabilities by wireless devices and sensors. Such systems require new, more sophisticated, event correlation over time and space. The integration of a smart WSN with a large network increases its coverage and potential application domain. In WSNs, a sink node is a sensor node with gateway functions to link to external networks such as the Internet. Sensed information is normally distributed via a sink node. A sink node may also be an gateway node to an intermediate ad hoc network, which may deliver the sensor data to the Internet. Fig.1 depicts WSNs connecting to the Internet, where an ad hoc network conveys sensed data to the Internet.

This new platform enables the seamless use of the various resources in physically interacting environments. A consensus is emerging that the most appropriate system architecture to support such platforms is service management, with communication based on the publish/subscribe paradigm. For example, a publisher broker node can act as a gateway from a WSN, performing data aggregation and distributing filtered data to other networks based on contents. Event broker nodes that offer data aggregation services can coordinate data flow efficiently. Especially when event-based communication is implemented via a peer-to-peer (P2P) overlay network, the construction of event broker grids will extend the seamless messaging capability over scalable heterogeneous network environments. Event Correlation will be a multi-step operation from event sources to the

final subscribers, combining information collected by wireless devices into higher level information or knowledge. Mobile devices can be deployed in remote locations without a network infrastructure.

In existing middleware and applications, the semantics of operators for composite events is not defined in a uniform manner leading to a number of problems. Event consumption rules are mostly done as part of an implementation, without a clear semantic definition. Most extant approaches to define event correlation lack a formal mechanism to define complex temporal and spatial relationships among correlated events. Thus, a unified semantics has to be defined to resolve this ambiguity. Temporal ordering in real-time is a critical aspect of event correlation in wireless ad hoc network environments. Neither logical time nor classical physical clock synchronization algorithms may be applicable. In order to determine the direction of movement of a real world entity, temporal ordering of events originating from different devices has to be established. Events can be triggered by physical phenomena, such as glaciers and earthquakes, and the order of occurrence of sensed data is again important.

Event Aggregation, Filtering and Correlation: Some event-based middlewares offer content-based filtering and provide flexible query languages. These allow subscribers to select events of interest, based on the values of their contents. A query can apply to different event types but the aim is to select individual events. On the other hand, event correlation addresses the relationship among, or pattern of, instances of different event types. WSNs have led to new issues to be addressed in event correlation. In WSNs the requirement is to summarize current sensor values in some or all of a sensor network. TinyDB [10] is an inquiry processing system for sensor networks and takes a data centric approach. Each node keeps the data and executes retrieval and aggregation (in-network aggregation), with on-demand based operation to deliver the data to external applications. TinyLIME [3] enhances LIME (Linda In Mobile Environments) to operate on TinyOS. In TinyLIME, LIME is maintained on each sensor node together with a partition of a tuple space. A coordinated tuple space is created across the nodes, connecting with the base station in one hop. It does not currently provide any data aggregation function, only a data filtering function based on Linda/LIME at the base station node. On the other hand, TinyDB supports data aggregation via SQL query, but redundancy/duplication handling is not clear from available documents.

Middleware research for WSNs has been active recently, but most research focuses on in-network operation for specific applications. In this paper, we take a global view of event correlation over entire distributed systems. We define generic correlation semantics, combining traditional event composition and data aggregation in wireless sensor networks. For the event detection semantics, we introduce a parameterized algebra. Parameters include time, selection, consumption, and subset rules. This approach defines

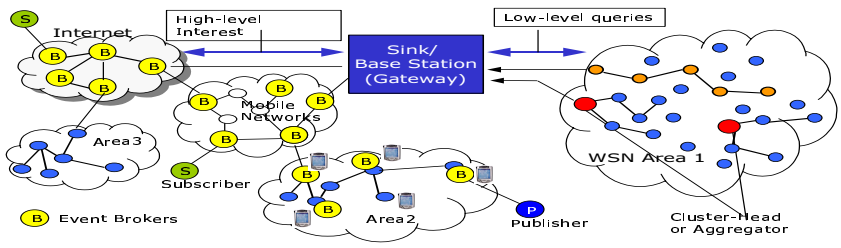

Fig. 1: Bridging WSNs to the Internet

unambiguous semantics of event detection and supports resource constrained environments. We introduce interval-based semantics for event detection defining precisely complex timing constraints among correlated event instances. In resource constrained network environments, the event algebra must be restricted so that only a subset of all possible occurrences of complex events will be detected, and this can be achieved by applying appropriate parameters. This paper continues as follows: Section 2 describes the event model, Section 3 defines composite event semantics and section 4 discusses temporal ordering. The formal definition and proof of the event algebra is out of the scope of this paper. Section 5 presents results of experiments. In Section 6 we describe related work, and Section 7 contains conclusions and directions for future research.

2 Event Model

In this section, we define our event model. A primitive event is the occurrence of a state transition at a certain point in time. Each occurrence of an event is called an event instance. Each event has a timestamp associated with the occurrence time. Composite events are defined by composing primitive or composite events with a set of operators.

Timestamps: A timestamp is a mandatory attribute of an event defined within a time system, while the event occurrence time is a real-time defined by the occurrence of the event. Thus, the timestamp is an approximation of the event occurrence time. Most point-based timestamps consist of a single value indicating the occurrence time. In [7], the time when an event is detected is given as an interval-based timestamp, which captures clock uncertainty and network delay with two values: the low and high end of the interval. Although an interval format is used, it represents a single point (point-interval-based timestamp). In addition, we define a (composite) event with duration and give a new interval-based timestamp to a composite event based on interval semantics. A point-interval-based timestamp is an accurate representation, and it is distinct from interval-based timestamps representing the duration of events.

Spacestamp: We introduce spacestamp, as an optional attribute of an event, indicating certain location, relative location, grouping and so forth (e.g., position information (latitude, longitude, elevation) by GPS, global node id). This information can be used for ordering events within the given space.

2.1 Duration

Reference [1] argues that all events have duration and considers intervals to be the basic timing concept. A set of 13 relations between intervals is defined, and rules governing the composition of such relations controls temporal reasoning. On the other hand, most event systems consider events as instantaneous, that is, the time associated with the event is an instant rather than an interval. A durative event can be seen as capturing the uncertainty over the time of occurrence and the time of detection of an event rather than modelling an event that persists over time. In this sense, durative events are akin to the point-interval-based-timestamps described above. The durative event model considers that instantaneous events are durative events with minimum duration, thus reconciling the models. We would regard an ‘event’ that persists over time as akin to a state, with an event at the start and one at the end of the time period. This could also be defined as a composite event. Composite events are built up from events occurring at different times, therefore the associated real-time is usually that of the last of its contributory primitive events. This is natural in a context where the prime focus is on event detection, since

typically a composite event will be detected at the time that its last contributory event is detected. However, this does lead to logical difficulties in the case of some composite events. Determination of the duration of composite events requires the semantics of composition and time system information such as a point-based or an interval-based time model.

2.2 Duplication

It is important to distinguish between multiple instances of a given event type and duplicates of a given event instance. The expressiveness of some event specification languages has been limited by not distinguishing between event types and instances of those types. [8] attempts to define conditions and constraints on attributes of events in correlation rules rather than defining operators on event instances. Especially in sensor networks, in order to avoid loss of events by communication instability, duplicates of events may be produced to increase reliability. Duplicates have to be handled differently depending on the application, and contexts within applications. For example, in object tracking, the most recent reading from a sensor is valid, and events prior to that will be obsolete, except for the historical record. On the other hand, in a transaction event in which a customer cancels an order, a duplicate event should be ignored because a transaction is being repeated. Thus, the semantics of event composition have to address handling of duplicates. [9] take the approach of defining constraints on attributes of events and detect occurrences of events, before correlation conditions are evaluated. We propose duplicate handling in two ways: adding a selection operator as an event composition operator and adding subset policies as parameters.

3 Event Correlation Semantics

We define composite events by expressions built from primitive and composite events and algebraic operators. The operators of the event algebra are defined informally in this section. We also support parameters, which help to define unambiguous semantics of event detection and support resource constrained environments. In wireless ad hoc networks, the event algebra must be restricted so that only a subset of all possible occurrences of complex events will be detected. We provide basic operators that have the potential of expressing the required semantics and are capable of restricting expressions. Also, an interval semantics supports more sensitive interval relations among events in environments where real-time concerns are more critical, such as wireless networks or multi-media systems. The temporal operators introduced in [1] are not uniformly defined in many applications. We define complex timing constraints among correlated event instances. An example is shown in Fig.2 (see full definition in [16]).

Relation	Timestamps of Primitive Events	Point	Interval
A before B	P-P: $t_p(A) < t_p(B)$ I-I: $t_i(A)^h < t_i(B)^l$	○ A ○ B	○–A–○ ○–B–○
(A + B)		●–●	●——●
(A | B)		●	●—●
(A ; B)		●–●	●——●
A overlaps B	P-P: NA I-I: $(t_i(A)^l < t_i(B)^l) \wedge (t_i(A)^h > t_i(B)^l)$		○—-A—-○ ○—-B—○
(A + B)			●——●
(A | B)			●——●
(A B)			●——●

Fig. 2: Temporal Condition for Composite Events

3.1 Composite Event Operators

The event operators are defined informally as follows:

- **Conjunction A + B:** Event A and B occur in any order.
- **Disjunction A | B:** Event A or B occurs.
- **Concatenation A B:** Event A occurs before event B where timestamp constraints are *A meets B, A overlaps B, A finishes B, A includes B, and A starts B.*
- **Sequence A ; B:** Event A occurs before B where timestamp constraints are *A before B, and A meets B.* (A ; B)$_T$ denotes that an interval T between event A and B.
- **Concurrency A||B:** Event A and B occur in parallel.
- **Iteration A*:** Any number of event A occurrences.
- **Negation $-\mathbf{A}_T$:** No event A occurs for an interval T. $(A; B) - C$ denotes that event A is followed by B and there is no C in the duration of (A;B).
- **Selection $\mathbf{A}^N$:** The selection A^N defines the occurrence defined by N. A_T^{AVG} denotes taking the average during an interval T.
- **Spatial Restriction $\mathbf{A}_S$:** Event A occurs if it is a spatial restriction defined in S, that can be defined as a specific location or a group identifier etc.
  - E.g. A_{CB03FD}: The area code *CB03FD* identifies the zone around Computer Laboratory in Cambridge. Event A is valid only when spatial condition is satisfied.
- **Temporal Restriction $\mathbf{A}_T$:** Event A occurs within T. B_T denotes a valid interval for B.

Example: The temperature of rooms with windows facing south is measured every minute and transmitted to a computer placed on the corridor. T denotes a temperature event and T_{30}^{AVG} denotes a composite event of an average of the temperature during 30 minutes. $(T_{room1} + T_{room7})_{30}^{AVG}$ denotes to take an average of room 1 and 7.

3.2 Interval Semantics

We give a timestamp to a composite event based on interval semantics. In most event algebras, each event occurrence, including composite events, is associated with a single time point. This may result in unintended semantics for some operator combinations, for example nested sequence operators. In Fig.3, time flows from left to right, and each row shows the occurrence of a primitive event. When single point detection is used, an instance of event B;(A;C) is detected if A occurs first, followed by B and C. The reason is that these occurrences cause a detection of A;C, which is associated with the occurrence of B;(A;C). With interval semantics, the sequence A;B can be defined to occur only if the intervals of A and B are non-overlapping. No occurrence of B;(A;C) would be detected.

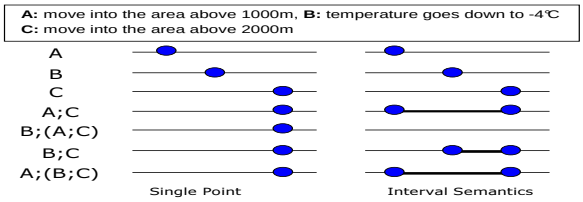

Fig. 3: Point and Interval

3.3 Event Context

Adding the policy defining the constraints provides a way to modify the operator semantics. This parameter-dependent algebra can accommodate different policies on event

consumption. First, each operator is given a principle definition of the constraints on the participating occurrences of events that characterize the operator. Then a number of event contexts are defined that act as modifiers to the simple operator semantics. These contexts specify constraints on how occurrences may be selected. As a result, each combination of an operator and a context can be seen as a separate operator with a specific meaning.

Consumption Policy: Three event consumption policies can be defined; *unrestricted*, *recent* and *chronicle*. Snoop [2] uses these contexts, but it is not capable of applying an individual context to different event operators. The parameter dependent algebra clarifies the situations. The following gives an informal definition for detecting A;B.

- Unrestricted: All instances of A and B are valid.
- Recent: If an instance of B can be combined with several instances of A to form instances of A;B, the only recent instance of A is valid.
- Chronicle: Only the oldest instance of A is valid, which is never valid in the future.

Subset Policy: defines the subset of events to detect. Ideally the *Subset Policy* should interfere as little as possible with unrestricted semantics. None of the removed instances should have a crucial impact on the detection of an enclosing detection. At the same time, operations such as conjunction and sequence must be able to identify non-valid instances early, before the end time of the instance is reached. The main task of the *Subset Policy* is to make an effective algebra, feasible to implement in resource constrained environments. The basis of the *Subset Policy* is that the restricted event stream should be a subset that does not contain multiple instances with the same end time.

Precision Policy: defines the precision of the events to be detected. The dynamic spatial-temporal data from WSNs is generated at a rapid rate and all the generated data may not arrive at the aggregation node over the networks due to the lossy/faulty nature of the sensor network. On the other hand, if some imprecision of the collected data could be tolerated by the application, defining the precision available is important.

For example, *High, Default, Low* can map to:

- the ratio of sensor nodes that are awake: *80%, 20%, 5%*
- the delivered time-series data: *100%, 70%, 50%*
- the interval of data collection: *1 second, 10 seconds, 60 seconds*
- the frequency of data report: *Urgent, Periodical, Available.*

3.4 Event Detection

The current detection mechanism is based on an imperative algorithm, which is executed once every time instant. The main loop selects subexpressions dynamically and computes the current instance of a target composite event from the current event and stored past information. For example, E denotes the event expression to be detected, and subexpressions of E are indexed 1 to k in bottom-up order. The operation result is $E^k(= E)$. Each operation in the expression needs its own indexed state variables (e.g., past events, time instant, and spatial information). A canned detection component can be created for common use. We implemented a prototype using a simple automata with support of parameterized values and time constraints.

4 Temporal Ordering

Sensor networks are used to monitor real world phenomena and for such monitoring applications, physical time is crucial. In global computing environments, such sensor data

flow over heterogeneous networks. We cannot assume a global clock, or globally synchronized physical clocks, to correlate events. Moreover when the store-and-forward paradigm is used for communication, message propagation delay is unavoidable. Traditional message ordering based on a transport layer protocol is not applicable. Thus, we use timestamps embedded in events for correlation, which provide a real-time mechanism. Temporal ordering of events is highly influenced by the event detection method, timestamping methods and the underlying time systems.

In many real world scenarios, wireless networks may be deployed with relay nodes to the Internet and it is possible that relay nodes can connect to Global Positioning System (GPS). GPS may be the key for providing accurate time adjustment at certain nodes that are less resource constrained within wireless ad hoc networks. We define two categories of network environments; where NTP is deployed with GPS, such as the Internet domain, and where networks are isolated in ad hoc mode without GPS or any other deterministic time synchronization mechanism. For the first category, we use interval-based point timestamps for primitive events, where the interval low and high end values are computed as described in [7] to allow for clock uncertainty and network delay. For timestamping composite events we use interval-based semantics, unlike[7] where a new timestamp is taken on the detection of a composite event. For the second category, several time synchronization mechanisms have been proposed. Among those, we investigated the one described in [14]. The idea of the algorithm is not to synchronize the local computer clocks of the devices but instead to generate timestamps from a local clock. When such locally generated timestamps are passed between devices, they are transformed to the local time of the receiving device. We propose a simplified protocol *Lightweight Local Clock Propagation*, which is on-demand based timestamp synchronization. The basic idea is that each node calculates its processing time using a local clock, and at the sink node, the sum of the processing times is subtracted from the event arrival time to estimate the occurrence time. Comparable timestamps are therefore created at sink nodes instead of network-wide. This requires the two assumptions: network delay is negligible (e.g. the node is close to the radio or network deployment is dense) and clock drift is negligible (e.g. the node carries an oscilloscope that guarantees less than 10 ppm drift).

Thus our proposal is a coordinated approach with and without the use of GPS. Sensor events could be aggregated at gateway nodes with transformed timestamps and passed towards a subscriber node in the Internet environment, where GPS-based time synchronization is deployed.

5 Experiments

Below are experiments results for composite event detection with time restriction, subset rule, and an object tracking system. Surface meteorological data from the NOAA Aeronomy Laboratory TRMM profile system recorded in 1999 are used as base data for the experiments described in Section 5.1-2. The data contain wind, temperature, relative humidity, pressure and solar radiation. The data from each instrument were sampled every 0.5 second. Every 10 seconds, a 10 second average was transmitted to a base station. As part of the 10 second average, a timestamp was added to the data. The clock is kept in UT time, and is set to the GPS time standard every week. The logged data were assembled to the individual event and a discrete event generator simulates

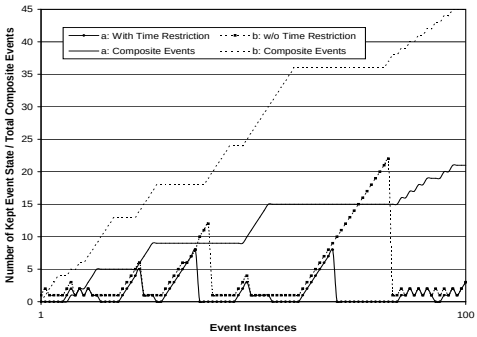

Fig. 4: Time Restriction

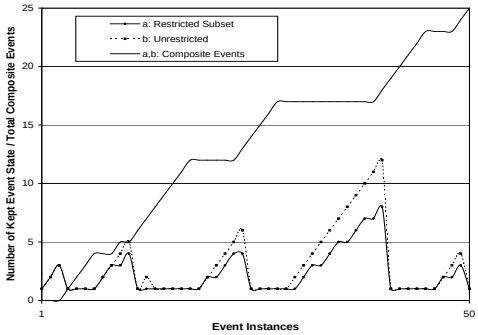

Fig. 5: Subset Rule

the event occurrences as if sensed data are reported from the sensor network in single hop communication range. In the first experiment, each event is duplicated. There is no loss of events in both experiments. The composite event used in the experiments is as follows: humidity below 60% (H) followed by temperature over 30% (T) within 10 seconds, which denotes $(H;T)_{10}$.

5.1 Time Restriction

Consider as an example, the event expression $(H;T)_4$: before the current time instant 10, there have been six occurrences of H. If a T_4 instance occurs in the current time instance, and the start time of this instance is 9 or 10, it should be combined with h_6 (occurred at time instant 7) to form an instance of $H;T_4$. If the start time is 8, it should be combined with h_5 (at 6), etc. Since an instance of T_4 with an end time of 10 must start no earlier than 6, it follows that it must be combined with either h_3 (at 4), h_4 (at 5), h_5, or h_6, and thus there is no need to store h_1 (at 1) and h_2 (at 2). Throughout the detection of this expression, all instances of H that end more than 4 time units ago, except the one with the latest start time, can be discarded. Fig. 4 shows a simulation of memory usage with the number of event states to be kept. For the detection policy, the most recent instance of H is used. The time restriction value is set to 10, which may be relatively a large number. The time restriction is a similar concept of event detection with a sliding window, but with our approach, semantics are unambiguous and capability of individual definition for each event gives another advantage. Our approach ensures that detection of composite events can be efficiently implemented with limited resources, which can be an critical element for embedded applications. The number of detected composite events is also illustrated in Fig. 4 that shows only half of composite events being detected when time restriction is specified. If undetected composite events imposes loss of information, then the time restriction number should be reduced.

5.2 Subset Rule

Fig. 5 illustrates the memory usage when the H's subset H_S. The event instances with the same end time is removed and the Subset Rule keeps exactly one with maximal start time. Because the composite event $(H_S;T)$ is equivalent to $(H;T)$, in an environment with the Subset Rule all the composite events, supposed to be detected in unrestricted environments should be detected. In Fig. 5, both cases show the same results for composite event detection. On the other hand, with the Subset Rule less states are kept.

5.3 Object Tracking

We developed a prototype of an object tracking system using the Active BAT data (see [17] for more detail). Fig.6 shows the specific period, when two people are positioned

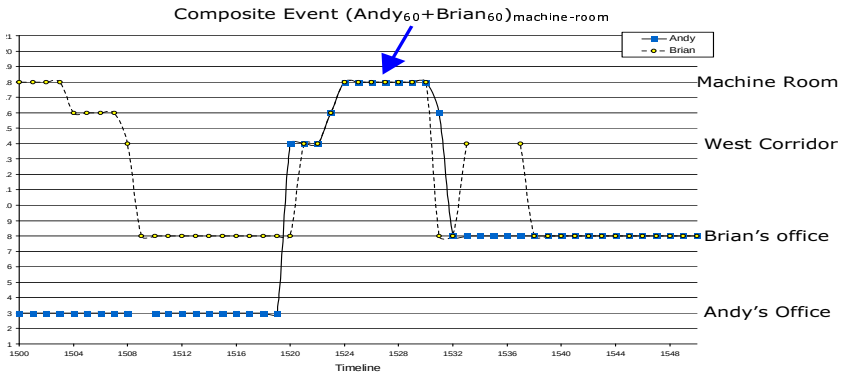

Fig. 6: Composite Event Detection

at the machine room. The composite event *(Brian+Andy)machine-room*, where 'Brian and Andy are together at the machine room' is detected between the time unit 1523 and 1529.

6 Related Work

Much composite event detection work has been done in active database research. SAMOS [5] uses Petri nets, in which event occurrences are associated with a number of parameter-value pairs. The transition from centralized to distributed systems led to the need to deal with time. [2] presents an event-based model for specifying timing constraints to be monitored. [9] proposes an approach that uses the occurrence time of various event instances for time constraint specification. GEM [11] allows additional conditions, including timing constraints, to combine with event operators for composite event specification. Composite events service in an event-based middleware system reduces the communication within the system and potentially gives a higher overall efficiency, which is addressed in [12]. Hayton et al. [6], on composite events in the Cambridge Event Architecture, describe an object-oriented system with an event algebra that is implemented by nested push-down FSA to handle parameterized events.

Temporal message ordering has been an issue in traditional networks such as for system monitoring and in distributed event systems. In existing systems, the semantics of event order often depends on the application logic. For real-time support, a common solution in wired networks provides a virtual global clock that bounds the value of the sum of precision and granularity within a few milliseconds. The 2g-Precedence model is enhanced for distributed event ordering and composite event detection using 2g-precedence-based sequence and concurrency operators [15]. However, in open distributed environments, not all servers are interconnected and event ordering based on NTP may lead to false event detection. Interval-based time systems define event order based on intervals. In [7], timestamps of events can be related to UTC (Universal Coordinated Time) with bounded accuracy, and event timestamps are modeled using accuracy intervals. They use NTP that provides reference time injected by a GPS time server and, in addition, returns reliable error bounds. For wireless network environments, [13] presents a GPS based virtual global clock, which is used for timestamping events, and deploys a similar concept to 2g-precedence. Post-facto synchronization [4] is based on unsynchronized local clocks but limits synchronization to the transmit range of the mobile nodes.

7 Conclusions and Future Work

Our event correlation semantics supports a new paradigm coming from the recent evolution of ubiquitous computing with a rapid increase of event monitoring capability by wireless devices. The main focus is on supporting time and space related issues such as temporal ordering, duplicate handling, and interval-based semantics, especially for wireless network environments. Event management will be a multi-step operation from event sources to final subscribers, combining information collected by wireless devices into higher-level information or knowledge in a global computing environment. Work is ongoing on the transformation of event algebra, so that complex expressions can be more efficiently implemented in resource constrained devices over wireless ad hoc networks. We are working on a complete implementation, including various timestamping environments.

Acknowledgment. This research is funded by EPSRC (Engineering and Physical Sciences Research Council) under grant GR/557303.

References

1. Allen, J. et al. Maintaining Knowledge about Temporal Intervals. *CACM*, 26(11), 1983.
2. Chakravarthy, S. et al. Snoop: An expressive event specification language for active databases. *Data Knowledge Engineering*, 14(1), 1996.
3. Curino, C. et al. TinyLIME: Bridging Mobile and Sensor Networks through Middleware. *Proc. PerCom*, 2005.
4. Elson, J. et al. Wireless Sensor Networks: A New Regime for Time Synchronization. *Workshop on Hot Topics in Networks (Hotnets-I)*, 2002.
5. Gatziu, S. et al. Detecting Composite Events in Active Database Systems Using Petri Nets. *Proc. RIDE-AIDS*, 1994.
6. Hayton, R. et al. OASIS: An Open architecture for Secure Inter-working Services. *PhD thesis, Univ.of Cambridge*, 1996.
7. Liebig, C. et al. Event Composition in Time-dependent Distributed Systems. *Proc. 4th CoopIS*, 1999.
8. Liu, G. et al. Composite Events for Network Event Correlation. *Proc. IFIP/IEEE International Symposium on Integrated Network Management*, 1999.
9. Liu, G. et al. A Unified Approach for Specifying Timing Constraints and Composite Events in Active Real-Time Database Systems. *Proc. RTAS*, 1998.
10. Madden, S. et al. TAG: A tiny aggregation service for ad-hoc sensor networks. *Proc. of Operating Systems Design and Implementation*, 2002.
11. Mansouri-Samani, M. et al. GEM: A Generalized Event Monitoring Language for Distributed systems. *IEE/IOP/BCS Distributed systems Engineering Journal*, 4(2), 1997.
12. Pietzuch, P. Shand, B. and Bacon, J. Composite Event Detection as a Generic Middleware Extension. *IEEE Network Magazine*, 18(1), 2004.
13. Prakash, R. et al. Causality and the Spatial-Temporal Ordering in Mobile Systems. *Mobile Networks and Applications*, 9(5):507-516, 2004.
14. Roemer, K. Time Synchronization in Ad Hoc Networks. *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc01)*, 2001.
15. Schwiderski, S. Monitoring the Behavior of Distributed Systems. *PhD thesis, University of Cambridge*, 1996.
16. Yoneki, E. and Bacon, J. Unified Semantics for Event Correlation over Time and Space in Hybrid Network Environments. *Proc. CoopIS*, 2005.
17. Yoneki, E. and Bacon, J. Object Tracking using Durative Events. *Proc. NCUS*, 2005.