Unified Semantics for Event Correlation over Time and Space in Hybrid Network Environments

Eiko Yoneki and Jean Bacon

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

Abstract. The recent evolution of ubiquitous computing has brought with it a dramatic increase of event monitoring capabilities by wireless devices and sensors. Such systems require new, more sophisticated, event correlation over time and space. This new paradigm implies composition of events in heterogeneous network environments, where network and resource conditions vary. 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. Most extant approaches to define event correlation lack a formal mechanism for establishing complex temporal and spatial relationships among correlated events. Here, we will focus on two subjects. First, 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. Secondly, we introduce interval-based semantics for event detection, defining precisely complex timing constraints among correlated event instances.

1 Introduction

An event correlation service is important for constructing reactive distributed applications. It occurs as a part of applications, event notification services and workflow coordinators. 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.

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. Typical Wireless Sensor Networks (WSNs) communicate directly with a centralized controller or a satellite. Thus, communication between a sensor and a controller is based on a single-hop model. On the other hand, a collection of autonomous nodes or terminals may communicate with each other by forming a multi-hop radio network and maintaining connectivity in a decentralized manner, thus creating an ad hoc network. Moreover, the integration of a smart WSN with a large network such as the Internet, increases its coverage and potential application domain. In WSNs, a sink node is a sensor node with

R. Meersman and Z. Tari (Eds.): CoopIS/DOA/ODBASE 2005, LNCS 3760, pp. 366–384, 2005.

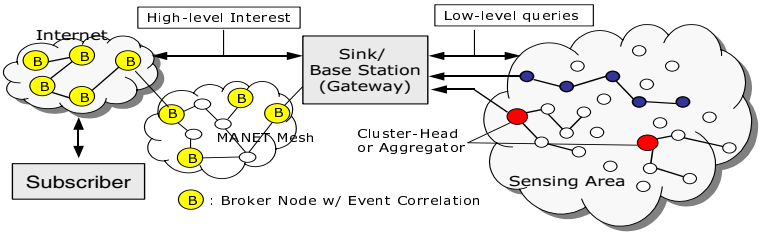

Fig. 1. Bridging WSNs to the Internet

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 Internet backbone node, or the 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 (the MANET Mesh in Fig.1) could form an opportunistic network such as [3].

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. They will have an important role in collecting sensor data over ad hoc networks and conveying it to Internet backbone nodes, These mobile devices are resource constrained, and an implementable event detection mechanism is required. The semantics of operators for composite events is not defined in a uniform manner in existing middleware and applications 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 for heterogeneous network environments.

Temporal ordering in real-time is a critical aspect of event correlation in wireless ad hoc network environments. The context of real-time in this paper relates to real-world event occurrences. Neither logical time nor classical physical clock synchronization algorithms may be applicable. Temporal ordering is required for motion detection. 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. In general, recent developments in Internet business

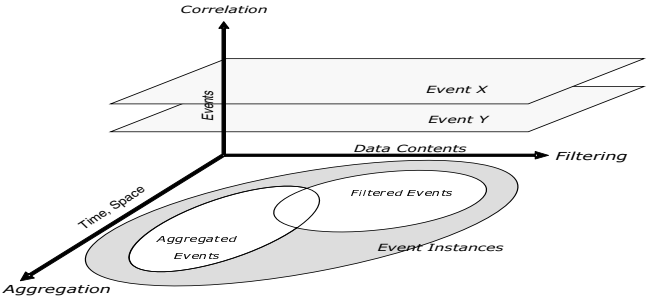

Fig. 2. Aggregation, Filtering and Correlation

demand precision time for processing data. For example, orders to buy and sell on the stock market and in an auction must be timed on a global scale, and a trigger event may be initiated by a wireless devices.

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. Traditional networking research approached data aggregation as an application-specific technique that can be used to reduce network traffic. In WSNs the requirement is to summarize current sensor values in some or all of a sensor network.

TinyDB [19] 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 [5] 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. TinyLIME works as middleware by offering this abstracted interface to the application. 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. Fig.2 shows the relationships between aggregation, filtering and correlation. 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, focusing on correlation semantics.

Unified Semantics for Event Correlation: We define a unified semantics, combining traditional event composition and data aggregation in wireless sensor networks. In-network aggregation in WSNs is not a concern of this paper. For the event detection semantics, we introduce a parameterized algebra. Parameters include time, selection, consumption, and subset rules. This approach defines 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 key aspects of event composition semantics with existing systems as examples. Section 3 describes the event model, Section 4 defines composite event semantics and section 5 discusses temporal ordering. The formal definition and proof of the event algebra is out of the scope of this paper. In Section 6 we describe related work, and Section 7 contains conclusions and directions for future research.

2 Event Correlation in Middleware

In this section, we show a comparative study of existing event correlation mechanisms in middleware. Event correlation may be deployed as a part of applications, as event notification services, or as part of a middleware framework. Definition and detection of composite events vary, especially over distributed environments. Equally, named event composition operators do not necessarily have the same semantics, while similar semantics might be expressed using different operators. Moreover, the exact semantic description of these operators is rarely explained. Thus, we define the following schema to classify existing operators: conjunction, disjunction, sequence, concurrency, negation, iteration, and selection. Considering the analyzed systems, it becomes clear that to simply consider the operators is not sufficient in order to convey the full semantic meaning. Each system offers parameters, which further define/change the operators' semantics. The problem is that the majority of the systems reflect parameters within the implementations. Parameters for consumption mode and duplicate handling are rarely explicitly described. Table 1 and 2 shows a comparative study of event composition semantics in ten existing systems and our proposal ECCO. The tables give an overview of event-based composition languages typically supported in event-based systems, and provide an analysis of the languages from a unified viewpoint. Note that the list of analyzed systems cannot be exhaustive, but considers a representative set of selected composition semantics. None of the listed systems includes wireless network environments except ECCO. Most event notification systems still only support primitive events, and their focus is on efficient filter algorithms. Brief explanations of the characteristics of each system are given below (see also Section 6, related work).

Table 1 shows composition operators, while Table 2 emphasizes temporal order related parameters including consumption modes. Concepts of conjunction, disjunction, sequence, concurrency, negation, and iteration operators are based on an event algebra. Selection defines the event instance selection, by which events qualify for a composite event, and how duplicate events are handled. Conjunction and disjunction are supported in most systems, some of which implement the sequence operators (requiring ordering), fewer support negation. Selection and concurrency are rarely supported. Concurrency is difficult to determine for distributed systems. Time operators are not always supported, requiring a time handling strategy for distributed systems. Consumption mode

Table 1. Event Composition Semantics - Composition Operators

	Operators						
	Conjunction	Disjunction	Sequence	Conc.	Negation	Iteration	Selection
ECCO	$A+B$	$A\|B$	$A;B$	$A\|\|B$	$-A$	A^*	A^N
Opera	$A\|\|B$	$A\|B$	$A;BorAB$	-	$\neg A$	A^*	-
CEA	$A\&B$	$A\|B$	$A;B$	-	$-A$	-	-
Schwiderski	A,B	$A\|B$	$A;B$	$A\|\|B$	$NOTA$	A^*orA^+	-
A-mediAS	$A\&B$	$A\|\|B$	$A;B$	-	$-A$	-	$A^{[i]}$
Ready	$A\&\&B$	$A\|\|B$	$A;B$	-	$notA$	-	-
Eve	$CON(A,B)$	$DEX(A,B)$	$SEQ(A,B)$	$CCR(A,B)$	$NEG(A,B)$	$REP(A,n)$	-
GEM	$A\&B$	$A\|\|B$	$A;B$	-	$!A$	-	-
Snoop	A,B	$A\vee B$	$A;B$	-	-	A^*	-
Rebeca	$A\wedge B$	$A\vee B$			$\neg A$	A^*	-
SAMOS	A,B	$A\|B$	$A;B$	-	$NOTA$	$TIMES(n,A)$	$A^*/last(A)$

and temporal conditions are rarely made explicit. If they are explicit, several options are supported, otherwise they are hard-coded in the system and difficult to determine. The listed systems are notification services, event composition languages, and workflow coordinators in which common characteristics are fairly complete semantics of event composition.

ECCO: is our ongoing project to create an event brokering architecture for a distributed adaptive mobile environment [27]. Event correlation is part of event brokers, and grids of brokers are deployed over mixed network environments. The ECCO prototype is implemented with a MANET protocol as content-based publish/subscribe. This paper focuses on event correlation over event broker grids, which requires correlation of events from various network environments.
Opera: a framework for event composition in a large scale distributed system [23] aiming at reduction of event traffic by distributed composite event detectors. The language of composite events is based on FSA.
CEA: our early work on the Cambridge Event Architecture (CEA) extended the then-predominant, object-oriented middleware (CORBA and Java) with a *publish, register and notify* paradigm [11]. COBEA [21] is an event-based architecture used for the management of networks using CEA based composite event operators.
Schwiderski: enhanced the distributed event ordering and composite event detection by introducing 2g-precedence-based sequence and concurrency operators [26].
A-mediAS: an integrating event notification service that is adaptable to different applications specifically on handling composite events and event filtering methods [12].
Ready: an event notification service from AT&T Research similar to Siena. Ready supports composite events and its grouping functionality can be shared among clients [10].
EVE: combines characteristics of active databases and event-based architectures to execute event driven workflows [8].
GEM: GEM [20] is an interpreted generalized event monitoring rule based language.
Snoop: a model-independent event specification language [4], supporting parameter contexts. It supports temporal, explicit and composite events for active databases.

Rebeca: an event-based electronic commerce architecture focusing on event filtering in a distributed environment [22]. Temporal delays in event composition have been addressed in [16].
SAMOS: The SAMOS (Swiss Active Mechanism-based Object-oriented database system) [7] project addresses the specification of active behavior and its internal processing supporting Event-Condition-Action (ECA) rules. The detection of composite events is implemented based on colored Petri-Nets.

Table 2. Event Composition Semantics - Time-related Parameters

	Time Operator		Timestamp	Composition	Temp. Cond.	Consumption			Subset	Detection
	Period	Life				Unrest.	Recent	Chron.		
ECCO	$(A;B)_T, (A+B)_T$	$(A)_T$	P, PI, I	I	×	×	×	×	×	Algorithm
Opera	$(A,B)_T$	-	PI	P	-	-	-	-	-	T-FSA
CEA	-	-	P	P	-	×	-	-	×	FSA
Schwiderski	-	-	P	P	-	-	-	-	-	Rule
A-mediAS	$(A,B)_T, (A;B)_T, -A_T$	-	P	P	-	×	×	×	×	T-FSA
Ready	-	-	P	P	-	-	-	-	×	-
Eve	-	-	P	P	-	-	-	×	×	Graph
GEM	(A+timeperiod)	-	P	P	×	-	×	-	×	Rule
Snoop	$(A, [tiestring] : param, B)$	-	P	P	-	-	×	×	×	Graph
Rebeca	slide window	-	PI	P	-	-	×	×	-	Rule
SAMOS	-	-	-	-	-	-	-	×	×	Petri net

Time Operator: Period (between event A and B) Life (valid time for event A)
Timestamp: P (Point-based), PI (Point represented in Interval-based format), and I (Interval-based).
Composition: Event composition semantics. P (Point-based), and I (Interval-based).
Temporal Condition: Temporal conditions such as *A before B, A meets B, etc. for event composition.*
Consumption: Event consumption (Unrestricted, Recent, and Chronicle).
Subset: Parameters for selection or subset on duplication handling.
Detection: Implementation methods.

3 Event Model

In this section, we define our event model including timestamps for composite events. 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. The primitive event set contains all primitive events within the system. The event set consists of the set of primitive events and the set of composite events. Each event has a timestamp associated with the occurrence time. There is uncertainty associated with the values of timestamps in implemented systems. Composite events are defined by composing primitive or composite events with a set of operators. Only composite events are defined to have duration.

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 [16], 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 (see Section 4.3). For a primitive event, either a point-based or point-interval-based timestamp is used in our system. A point-interval-based timestamp is an accurate representation, and it is distinct from interval-based timestamps representing the duration of events. Fig.3 depicts an interval-based timestamp for composite events.

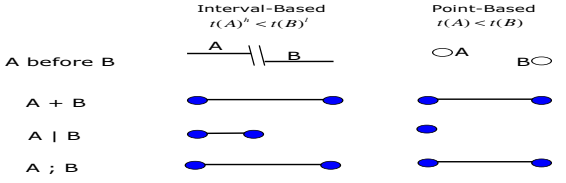

Fig. 3. Timestamp of Composite Event (A before B)

Spacestamp: We introduce spacestamp, as an optional attribute of an event, indicating certain location, relative location, grouping and so forth (e.g., position information (x,y,z), global node id). The Global Positioning System (GPS) [13] provides each node with its location information (latitude, longitude and elevation) with a high degree of accuracy. This information can be used for ordering events within the given space.

3.1 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 listed in Section 2 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. Our model is based on primitive events that represent instantaneous changes of system state, with uncertainty over their measurement. 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. Fig.4 shows an example of a sequence operation on the interval-based and point-based timestamps. Complex timing constraints among correlated event instances are precisely defined (see Appendix).

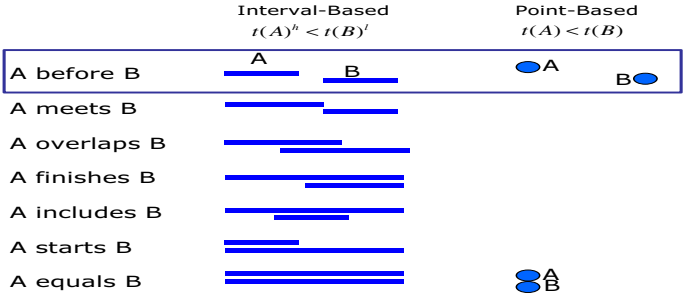

Fig. 4. Interval and Point based Timestamps

3.2 Duplication

It is important to distinguish between multiple instances of a given event type, which may be primitive or composite, 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. [17] 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. [18] 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 rules as parameters (obtaining this efficiently without loss of meaningful data) (see Section 4.4).

4 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. Our recent work [23] defined composite event operators that are tightly based on FSA without considering time-related issues in depth. The temporal operators introduced in [1] are not uniformly defined in many applications, and we define precise timing constraints (see Appendix and [28]). The temporal operators help resource constrained mobile devices to have unambiguous semantics.

4.1 Composite Event Operators

The event operators are defined informally as follows:

- **Conjunction A + B:** Event A and B occur in any order. (A + B)$_T$ with a temporal parameter T indicating the maximal length of the interval between the occurrences of A and B. Note that $(A + B)_\infty$ or $(A + B)$ refers without restrictions.
- **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)$_0$ is a special case belonging to *A meets B*.
  - E.g. (A;NULL;B): denotes there is no occurrence of any event between event A and B.
  - E.g. (A ; B)$_T$: means that an interval T between event A and B.
  - E.g. (A;B$_0$): denotes that there is event A and event B occur contiguously.
- **Concurrency A||B:** Event A and B occur in parallel.
- **Iteration A*:** Any number of event A occurrences.
- **Negation $-$A$_T$:** No event A occurs for an interval T.
  - E.g. $(A - B)$: denotes no B occurs during A's occurrence.
  - E.g. $(A - B)_T$: denotes no B occurs after starting A's occurrence within an interval T.
  - E.g. $(A; B) - C$: denotes that event A is followed by B and there is no C in the duration of (A;B).
- **Selection A^N:** The selection A^N defines the occurrence defined by N.
  - E.g. A_T^{AVG}: denotes taking the average during an interval T.
  - E.g. A_T^{LAST}: denotes taking the most recent instance during an interval T.
- **Spatial Restriction 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 A$_T$:** Event A occurs within T.
  - E.g. $(A; B)_T$ or $(A; B_T)$: B occurs within an interval T after A.
  - E.g. B_T: B is valid for an interval T.

The following examples illustrate the use of the operators to describe composite events.

Example 1: 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.

Example 2: Two sensing receivers are placed before and after the stop sign on the street. When a car passes, they generate events to the local computer. Suppose the event received before the stop sign is B and after the stop sign is A for a given car. $(B; A)_2$ denotes A occurs 2 seconds after B and it indicates a car did not make a full stop at the stop sign. On the other hand, $((B; A)_{60})^*$ indicates cars may not be flowing on the street, which indicates potential traffic congestion.

Example 3: At a highway entrance, a sensor detects movement of a passing car as event E. The number of cars entering the highway $(E_{10}^{SUM})_{HWY1Ent7}$ can be calculated at a computer locally, which can be used to detect traffic congestion on the highway (e.g., a congestion event C = $((E_{10}^{SUM})_{HWY1Ent7})^{LESS12}$).

4.2 Temporal Conditions

Defining temporal conditions for the semantics of composite events could be tricky, especially when timing constraints are important such as for processing transactions. This may cause an incorrect interpretation according to the intuition of the user. Fig.5

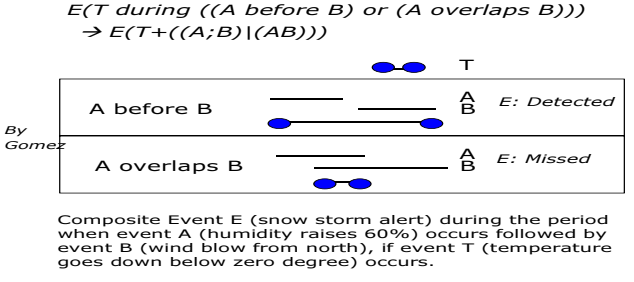

Fig. 5. Semantic Ambiguity

depicts a composite event E (*snow storm alert*): during the period when primitive event A (*humidity raises 60%*) occurs followed by primitive event B (*wind blows from north*), if primitive event T (*temperature goes down below zero degree*) occurs. Two situations are shown. If we follow the interpretation of temporal conditions described by Gomez et al. in [9], in the first situation, event E is detected and in the second situation, event E is missed. In [9], *overlaps* and *during* only comprises the period when two events are simultaneously occurring, while every other operator takes the period over both event

occurrences. This inconsistency may cause a problem. In both occasions, the natural interpretation of event E is the same. With our definition, both examples will yield consistent results.

4.3 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.6, 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.

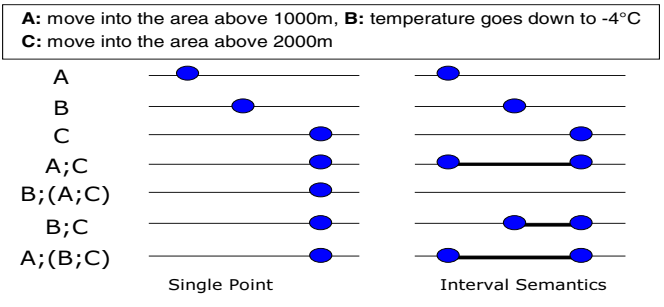

Fig. 6. Point and Interval

4.4 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. This helps to deal with resource constrained environments, e.g., keeping the most recent instance for future use.

Consumption Policy: For event consumption policy, three contexts *unrestricted*, *recent* and *chronicle*, can be defined. Snoop [4] 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. Fig.7 shows the effect of these contexts on the sequence operator.

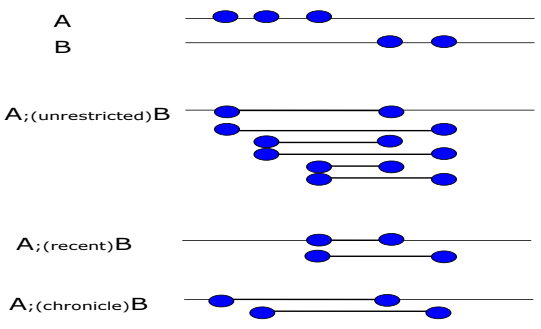

Fig. 7. Event Consumption Policy

- 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: If an instance of B can be combined with several instances of A to form instances of A;B, only the oldest instance of A is valid. Also, this instance 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.

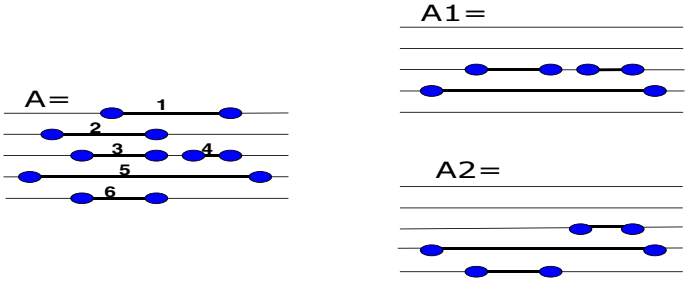

Fig. 8. Subset Policy

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. Informally, from the instances with the same end time, the *Subset Policy* keeps exactly one, that with the maximal start time. Fig. 8 shows the result of applying the *Subset Policy* to an event

stream A. From the three instances of A (2,3,6) with the same end time, 2 must be removed, together with either 3 or 6. For the two instances (1,4) that share the same end time, the one with the earlier time must be removed. 5 does not share the end time with others. The choice of which of (3,6) to remove results in two valid restrictions of the event stream A, named A1 and A2.

Precision Policy: defines the required precision of the events to be detected. The dynamic spatial-temporal data from sensor networks is generated at a rapid rate and all the generated data may not arrive at the event correlation node over the networks due to the lossy/faulty nature of the sensor network. Various techniques, including compression and model adaptation have provided certain levels of guarantees [15]. 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.*

4.5 Event Detection

The current detection mechanism is based on an imperative algorithm, which is executed once every time instant (Fig.9). The main loop selects subexpressions dynamically and computes the current instance of a target composite event from the current primitive 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). For example, *Negation* $E = (A; B) - C_T$, an instance of $(A; B)$ is an instance of $(A; B) - C$ unless invalidated by some instance of C. Thus, if the current instance of $(A; B)$ is valid, the instance of C with maximum start time can be the only one to invalidate. *Conjunction* requires storage of the instance with maximum start time from each subexpression. If the event expressions are known before the execution, a canned detection component can be created for common use. The formal proof of the algorithm is out of scope of this paper. We implemented a prototype based on a simple automata with support of parameterized values and time constraints.

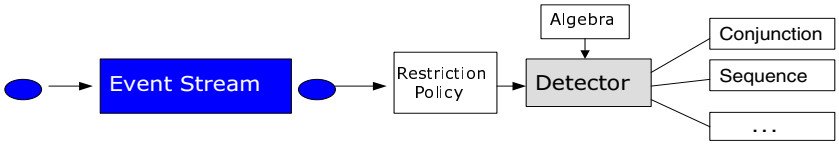

Fig. 9. Event Detection

5 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 and ordering events is difficult. 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.

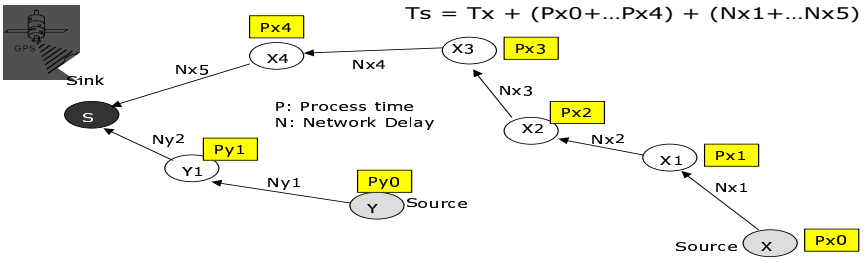

Fig. 10. Lightweight Local Clock Propagation

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 this section, we describe our time system for timestamping primitive events. Temporal ordering is based on interval semantics and is described precisely in the Appendix.

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 GPS. This makes us consider the use of GPS in distributed systems, including wireless network environments. However, GPS is not suitable for use in a large class of smart devices due to its high power consumption and the required line of sight to satellites. But 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 environment; 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 (see section 3), where the interval low and high end values are computed as described in [16] to allow for clock uncertainty and network delay. For timestamping composite events we use interval-based semantics, unlike[16] where a new timestamp is taken on the detection of a composite event. For the second category, several time synchronization mechanisms have been proposed (see section 6). Among those, we investigated the one described in [25]. 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. As a result of the experiments in [29], we propose a simplified protocol *Lightweight Local Clock Propagation* explained below.

Thus our proposal is a coordinated approach with and without the use of GPS. We use interval-based timestamps throughout, with interval-based point timestamps for primitive events and interval-based semantics for composite events. 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.

Lightweight Local Clock Propagation: is on-demand based timestamp synchronization. Fig.1 shows the operation from source nodes X and Y to the sink node S. 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 following two assumptions:

- Network delay is negligible, where the node is close to the radio or network deployment is dense. Thus, O(nanoseconds) over a path to the sink of O(100s) meters..
- Clock drift is negligible, where the node carries an oscilloscope that guarantees less than 10 ppm drift. One part per million (ppm) corresponds to 1 second in 11.5 days.

The detail of algorithms and experiments are out of the scope of this paper.

6 Related Work

Much composite event detection work has been done in active database research. SAMOS [7] uses Petri nets, in which event occurrences are associated with a number of parameter-value pairs. An early language for composite events follows the Event-Condition-Action (ECA) model and resembles database query algebras with an expressive syntax. Snoop [4] is an event specification language for active databases, which informally defines event contexts. The transition from centralized to distributed systems led to the need to deal with time. [4] presents an event-based model for specifying timing constraints to be monitored, and to process both asynchronous and synchronous monitoring of real-time constraints. [18] proposes an approach that uses the occurrence time of various event instances for time constraint specification. GEM [20] allows additional conditions, including timing constraints, to combine with event operators for composite event specification. In event-base middleware, publish/subscribe can provide subscription to composite events instead of leaving it to the client to subscribe to and correlate multiple primitive events. This reduces the communication within the system and potentially gives a higher overall efficiency, which is addressed in [23]. Hayton et al. [11], on composite events in the Cambridge Event Architecture [2], describe an object-oriented system with an event algebra that is implemented by nested push-down FSA to handle parameterized events. [12] provides time-restricted sequence and conjunction and a construct to detect the i^{th} occurrence of a given event.

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. Even the established Java Message Service (JMS) only guarantees the event order within a session where a session is a single-threaded context that handles message passing.

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 following approaches aim to support a real-time mechanism. The 2g-Precedence model is enhanced for distributed event ordering and composite event detection using 2g-precedence-based sequence and concurrency operators [26]. Events from different sites can only be ordered if they are at least two clock ticks apart. The 2g-precedence model is applicable for closed networks with interconnected servers. Network Time Protocol (NTP) is an Internet standard that supports the assignment of real-time timestamps with given maximal errors. 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 [16], 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. In [23], the interval timestamp model introduced in [16] is adopted.

For wireless network environments, [24] presents a GPS based virtual global clock, which is used for timestamping events, and deploys a similar concept to 2g-precedence. Without the existence of GPS, there is no means of synchronizing the clocks of all the nodes in a deterministic fashion with an upper bound independent of the message propagation delay and system size. Logical time cannot be used to determine temporal ordering, because causal ordering of events in the real world must be maintained. Thus, physical time has to be used, requiring clock synchronization. However, most of the synchronization algorithms rely on partitioned networks. Post-facto synchronization [6] is based on unsynchronized local clocks but limits synchronization to the transmit range of the mobile nodes. In [25], they take a similar approach of unsynchronized clocks.

7 Conclusions and Future Work

In this paper, we introduce a generic composite event semantics, which combines traditional event composition and the generic concept of data aggregation in wireless sensor networks. 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. Our approach includes definition of precise and complex temporal relationships among correlated events using interval-based semantics. 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, requiring more sophisticated event correlation over time and space. Work is ongoing on high level language definition and the transformation of event algebra, so that complex expressions can be more efficiently implemented in resource constrained devices over mobile ad hoc networks. We are working on a complete implementation, including various timestamping environments and parallel/hierarchical composition. Integration with WSN middleware such as TinyLIME is in progress.

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. Bacon, J. et al. Generic Support for Distributed Applications. *IEEE Computer*, 68-77, 2000.
3. Chaintresu, A. et al. Pocket Switched Networks: Real-world mobility and its consequences for opportunistic forwarding. *Technical Report, University of Cambridge*, 617, 2005.
4. Chakravarthy, S. et al. Snoop: An expressive event specification language for active databases. *Data Knowledge Engineering*, 14(1), 1996.
5. Curino, C. et al. TinyLIME: Bridging Mobile and Sensor Networks through Middleware. *Proc. PerCom*, 2005.
6. Elson, J. et al. Wireless Sensor Networks: A New Regime for Time Synchronization. *Workshop on Hot Topics in Networks (Hotnets-I)*, 2002.
7. Gatziu, S. et al. Detecting Composite Events in Active Database Systems Using Petri Nets. *Proc. RIDE-AIDS*, 1994.
8. Geppert, A. et al. Event-based distributed workflow execution with EVE. *Technical Report Univ. Zurich*, 1996.
9. Gomez, R. et al. Durative Events in Active Databases. *Proc. ICEIS*, 2004.
10. Gruber, B. et al. The architecture of the READY event notification service. *Proc. ICDCS*, 1999.
11. Hayton, R. et al. OASIS: An Open architecture for Secure Inter-working Services. *PhD thesis, Univ.of Cambridge*, 1996.
12. Hinze, A. et al. A flexible parameter-dependent algebra for event notification services. *Tech. Report, FU Berlin*, 2002.
13. Hoffmann-Wellenhof, B.H. et al. GPS: Theory and Practice. *Springer*, 1994.
14. Kopetz, H. et al. Real-time systems development: The programming model of MARS. *Proc. ISADS*, 1993.
15. Lazaridis, I. et al. Capturing Sensor-Generated Time Series with Quality Guarantees. *Proc. ICDE.*, 2003.
16. Liebig, C. et al. Event Composition in Time-dependent Distributed Systems. *Proc. 4th CoopIS*, 1999.
17. Liu, G. et al. Composite Events for Network Event Correlation. *Proc. IFIP/IEEE International Symposium on Integrated Network Management*, 1999.
18. Liu, G. et al. A Unified Approach for Specifying Timing Constraints and Composite Events in Active Real-Time Database Systems. *Proc. IReal-Time Technology and Applications Symposium*, 1998.
19. Madden, S. et al. TAG: A tiny aggregation service for ad-hoc sensor networks. *Proc. of Operating Systems Design and Implementation*, 2002.
20. Mansouri-Samani, M. et al. GEM: A Generalized Event Monitoring Language for Distributed systems. *IEE/IOP/BCS Distributed systems Engineering Journal*, 4(2), 1997.
21. Ma, C. et al. COBEA: A CORBA-based event architecture. *Proc. COOTS*, 1998.
22. Muehl, G. et al. Filter similarities in content-based publish/subscribe systems. *Proc. ARCD*, 2002.
23. Pietzuch, P. Shand, B. and Bacon, J. Composite Event Detection as a Generic Middleware Extension. *IEEE Network Magazine, Special Issue on Middleware Technologies for Future Communication Networks*, 2004.

24. Prakash, R. et al. Causality and the Spatial-Temporal Ordering in Mobile Systems. *Mobile Networks and Applications*, 9(5):507-516, 2004.
25. Roemer, K. Time Synchronization in Ad Hoc Networks. *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc01)*, 2001.
26. Schwiderski, S. Monitoring the Behavior of Distributed Systems. *PhD thesis, University of Cambridge*, 1996.
27. Yoneki, E. and Bacon, J. Distributed Multicast Grouping for Publish/Subscribe over Mobile Ad Hoc Networks. *Proc. IEEE WCNC*, 2005.
28. Yoneki, E. and Bacon, J. Determination of Time and Order for Event-Based Middleware. *Proc. PerCom-MP2P*, 2005.
29. Yoneki, E. and Bacon, J. Event Order with Interval Timestamp in Event Correlation Service over Wireless Ad Hoc Networks. *Companion Proc. Middleware*, 2004.

Appendix: Interval Semantics - Timestamp for Composite Events

Table 3. Interval Semantics - Timestamp for Composite Events

	Relation	Timestamps of Primitive Events	Point	Interval	Interval/Point	Point/Interval
1	A before B	P-P: $t_p(A) < t_p(B)$	○ A	○–A–○	○–A–○	○ A
		I-I: $t_i(A)^h < t_i(B)^l$	○ B	○–B–○	○ B	○–B–○
	(A + B)	I-P: $t_i(A)^h < t_p(B)$	●–●	●——— ——●	●——— ——●	●——— ——●
	(A | B)	P-I: $t_p(A) < t_i(B)^l$	● ●	●—— ——●	●—— ——●	●—— ——●
	(A ; B)		●–●	●——— ——●	●——— ——●	●——— ——●
2	A meets B *	P-P: NA		○—A–○	○—A—○	○ A
		I-I: $t_i(A)^h = t_i(B)^l$		○—B—○	○ B	○—B—○
	(A + B)	I-P: $t_i(A)^h = t_p(B)$		●——— ——●	●——— ——●	●——— ——●
	(A | B)	P-I: $t_p(A) = t_i(B)^l$		●—— ——●	●—— ——●	●
	(A B)			●——— ——●	●——— ——●	●——— ——●
	(A ; B)$_0$			●——— ——●	●——— ——●	●——— ——●
3	A overlaps B	P-P: NA		○—-A—-○		
		I-I: $(t_i(A)^l < t_i(B)^l) \wedge (t_i(A)^h > t_i(B)^l)$		○—-B—○		
	(A + B)	I-P: NA		●——— ——●		
	(A | B)	P-I: NA		●—— ——●		
	(A B)			●——— ——●		
4	A finishes B	P-P: NA		○———A———○	○—A—○	○ A
		I-I: $(t_i(A)^l < t_i(B)^l) \wedge (t_i(A)^h = t_i(B)^h)$		○—B—○	○ B	○—B—○
	(A + B)	I-P: $t_i(A)^h = t_p(B)$		●——— ——●	●——— ——●	●——— ——●
	(A | B)	P-I: $t_p(A) = t_i(B)^h)$		●——— ——●	●——— ——●	●——— ——●
	(A B)			●——— ——●	●——— ——●	
5	A includes B	P-P: NA		○———A———○	○———A———○	
		I-I: $(t_i(A)^l < t_i(B)^l) \wedge (t_i(A)^h > t_i(B)^h)$		○–B–○	○ B	
	(A + B)	I-P: $(t_i(A)^l < t_p(B)) \wedge (t_i(A)^h > t_p(B))$		●——— ——●	●——— ——●	
	(A | B)	P-I: NA		●——— ——●	●——— ——●	
	(A B)			●——— ——●	●——— ——●	
6	A starts B	P-P: NA		○—A—○	○—A—○	○ A
		I-I: $(t_i(A)^l = t_i(B)^l) \wedge (t_i(A)^h < t_i(B)^h)$		○———B———○	○ B	○—B—○
	(A + B)	I-P: $t_i(A)^l = t_p(B)$		●——— ——●	●——— ——●	●——— ——●
	(A | B)	P-I: $t_p(A) = t_i(B)^l$		●—— ——●	●——— ——●	●
	(A B)			●——— ——●		●——— ——●
7	A equals B	P-P: $t_p(A) = t_p(B)$	○ A	○———A———○		
		I-I: $(t_i(A)^l = t_i(B)^l) \wedge (t_i(A)^h = t_i(B)^h)$	○ B	○———B———○		
	(A + B)	I-P: NA	●	●——— ——●		
	(A | B)	P-I: NA	●	●——— ——●		
	(A || B)		●	●——— ——●		

●—● depicts the timestamp for the composite events
* A meets B where $t(A)^h$ and $t(B)^l$ share the same time unit
$t(A)$: timestamp of an event instance A
$t_p(A)$: Point-based timestamp
$t_i(A)_l^h$: Interval-based timestamp from event composition
$t_{pi}(A)_l^h$: Point-interval-based timestamp
(compared same as point-based timestamp but using interval comparison)

P-P: Between Point-based and Point-based timestamps
I-I: Between Interval-based and Interval-based timestamps
I-P: Between Interval-based and Point-based timestamps
P-I: Between Point-based and Interval-based timestamps

Real-time Period T:

$$[t(A)^l, t(A)^h] - [t(B)^l, t(B)^h] < T = \begin{cases} YES: & max(t(B)^h, t(A)^h) - min(t(B)^l, t(A)^l) \leq T(1- \\ NO: & max(t(B)^l, t(A)^l) - min(t(B)^h, t(A)^h) < T(1+ \\ MAYBE: & otherwise \end{cases}$$

where ρ is maximum clock skew.