Determination of Time and Order for Event-Based Middleware in Mobile Peer-to-Peer Environments

Eiko Yoneki and Jean Bacon
University of Cambridge Computer Laboratory
J J Thomson Avenue, Cambridge CB3 0FD, UK
Email: {eiko.yoneki, jean.bacon}@cl.cam.ac.uk

Abstract

An event correlation is becoming an important service in event-based middleware allowing subscribers in publish/subscribe paradigm to consume patterns of events (composite events). Recent evolution of wireless networks makes events flow from tiny sensor networks to Internet scale peer-to-peer systems among event broker grids. This new paradigm requires composition of events in heterogeneous network environments, where time synchronization and network conditions vary. Most extant approaches to define event correlation lacks a formal mechanism to define complex temporal relationships among correlated events. Here, we introduce generic composite events semantics introducing interval-based semantics for event detection supporting resource-constrained environments. We precisely define complex timing constraints among correlated event instances. We discuss underlying time systems and outline real-time temporal event ordering.

1 Introduction

An event correlation service is becoming important in event-based middleware for constructing reactive distributed applications. Event correlation takes places as a part of applications, event notification services or workflow coordinators. In publish/subscribe systems, an event correlation service allows consumers to subscribe patterns of events (composite events). This gives additional dimension of data management, improvement of scalability and performance in distributed systems. Particularly in wireless ad hoc networks, it helps to simplify the application logic and to reduce its complexity by middleware services. An event correlation service combines the information collected by the individual devices into higher level information or knowledge. There is a diversity of large scale network environments, and messages flow from tiny sensor networks to Internet-scale peer-to-peer (P2P) systems. The publish/subscribe paradigm becomes powerful in such environments, when not all the nodes are exclusively in an ad hoc topology. In this case, some nodes may be connected to the Internet backbone or may be relay nodes for different networks. For example, a publisher broker node can act as a gateway from a sensor network, performing data aggregation and distributing filtered data to other mobile networks based on the contents. Event broker nodes that offer data aggregation services can coordinate data flow efficiently. Especially with the recent evolution of distributed event-based middleware over P2P overlay network environments, the construction of event broker grids will extend the seamless messaging capability over heterogeneous network environments including mobile P2P environments. Event correlation will be a multi-step operation from event sources to the final subscribers.

When designing real-time systems, the time triggered approach is expensive when the expected rate of primitive event occurrence is low. An alternative is to use an event triggered approach where the execution is driven by events. The context of real-time in this paper is the time of a real event occurrence. Time systems may be classified into two categories: logical time and real-time mechanisms. Logical time is based on the causal order [10], however it does not contain real-time information. One of the critical function in event correlation is temporal ordering of events. Recent progress on Internet business requires precision time for processing the data. For example, buy and sell orders of stock market and an auction are critical for timing in a global scale. Securing document needs to contain timestamps within a cryptographic certification. Aviation traffic control, multimedia synchronization for real-time teleconferencing, and distributed network gaming also require precise timing in distributed environments. Temporal ordering in real-time is a critical aspect for event correlation in wireless ad hoc network environments. For example, to detect the direction of movement of a real-world phenomenon, temporal ordering of events originating from different devices has to be determined. The event can also be triggered by real-world phenomena. Neither logical time nor classical physical clock synchronization algorithms may be useful.

Definition and detection of composite events in the existing systems vary, especially over distributed environments, and 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. Event consumption rules are mostly done as a part of specific implementation without clear semantic definition. It is important to provide a well-defined unambiguous semantics for detection of composite events.

In [8], an attempt to transform existing event notification services with an unified event correlation service is described. However the current attempt left out time related issues. Time model varies in different event systems and different network environments. A wireless network environment brings further challenging problems. When real-time constraints involve event correlation, semantic ambiguity may arise from the timing of multiple instances of the same event. Because of the nature of unstable network environments, multiple events on the same incident may be created, and it is necessary to deal with redundancy and duplication of events. How to solve the semantic ambiguity and formalize it is a difficult issue that has been tuckled, but there is no definite solution yet. We propose an unified semantics definition that focuses temporal aspects especially for wireless network environments. Event monitoring capabilities are dramatically increasing by evolution of sensor networks, and data management in event-based middleware over hybrid mixed network environments is crucial. Note that the coordination of nodes within sensor network is different from other wireless ad hoc networks. The group of nodes acts as an single unit of processors in many cases. Thus, a solution described in this paper does not address the data aggregation within the sensor networks.

This paper continues as follows: Section 2 describes event model, and section 3 discusses time model. Then section 4 defines composite event semantics. The formal definition and proof of the event algebra is out of scope of this paper. Section 5 briefly presents the result of experiments. In section 6, we describes related work, and section 7 contains conclusions and directions for future research.

2 Event Model

A primitive event is the occurrence of a state transition at certain point in time. Each event has a timestamp associating to the occurrence time, and all the occurrences of events can be ordered in global system of reference. We define that both primitive and composite events can have duration. Composite events are created based on defined event composition algebra. A durative event can be seen as an abstraction constructed over two instantaneous events, which bound its occurrence period instead of start-end instantaneous events. Considering the time of occurrence and the time of detection, these two times in primitive events usually meet. Determination of the duration of composite events requires semantics of composition and the time system information. Complex timing constraints among correlated event instances are defined precisely (See Table 1). For example, the disjunction operation of event A and B may detect A as a result of the event composition, and if a point-based time system is used, the timestamp of event A is maintained as a timestamp of the composite event, while the conjunction operation of A and B results duration of A and B as a timestamp of the composite event.

Timestamp is a mandatory attribute of an event defined within a time systems based on an internal clock, 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.

Both primitive and composite events are recurrent and an event may occur multiple times and simply applying event operators on events instead of specific event instances might cause semantic ambiguity. Duplicates have to be handled differently depending on the application logic. Duplicates of the events can be several different instances or purely duplications of the same event. For example, the object tracking, the most recent reading from the sensor is valid and prior to that event will be obsolete. On the other hand, the transaction event in which a customer cancels the order, if the transaction comes in twice, the duplicate event should be ignored. Thus, semantics of event composition has to address handling of duplications. In [12], they take an approach by defining constraints on attributes of events and detect the occurrence of events, before correlation conditions are evaluated. We propose duplication handling in two ways: first, adding a selection operator as event composition operator. Secondly, subset rules are added as a parameter (see Section 4).

3 Temporal Ordering

Temporal ordering of events is highly influenced by the event detection method, timestamping methods and underlying time systems. In general, ordering events in distributed applications is difficult without a global clock. Moreover an event broker grid follows the model of store-and-forward paradigm, and message propagation delay is unavoidable. Traditional message ordering based on transport layer protocol is not applicable. Thus, timestamps embedded in events should be used for correlation. 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. We briefly look at the existing time systems for real-time mechanism below.

Time Systems: For real-time support, a common solution in wired networks provide 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 real-time mechanism. 2g-Precedence model [9] is enhanced for distributed event ordering and composite event detection using 2g-precedence-based sequence and concurrency operators [17]. Events from different sites can only be ordered if they are at least two clock ticks apart (2g). The 2g-precedence model is applicable for closed networks with interconnected servers. Network Time Protocol (NTP) is an Internet standard for real-mechanism. The NTP allows for assignment of real-time timestamps with given maximal errors. Global and partial order of events can be obtained with a certain accuracy based on timestamps. 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 [11], timestamps of events can be related to the global reference time with bounded accuracy and event timestamps are modeled using accuracy intervals. They use NTP that provides reference time injected by GPS time server and in addition return reliable error bounds. Similar to 2g-precedence, the accuracy intervals depend on both local and global timestamps but are created using NTP. The detection interval is supposed to equal to the accuracy interval.

For wireless networks environments, [15] presents GPS based virtual global clock that is used for timestamping events, and deploys the similar concept 2g-precedence. Without the existence of GPS, there is no mean to synchronize 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 obtained. Thus, physical time has to be used requiring clock synchronization. However, most of the synchronization algorithms rely on partitioned networks. Post-facto synchronization [4] is based upon unsynchronized local clocks but limits synchronization to the transmit range of the mobile nodes. In [16], they takes a similar approach of unsynchronized clocks. 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. The algorithm enables all participating nodes to reason about sets of timestamps received from arbitrary nodes and is not affected by topology changes. The detail of temporal ordering mechanism is out of scope of this paper.

Timestamping: Most of point-based timestamp consists of a single value indicating the point of time. In [11], the time when an event is detected is given as a new interval-based timestamp, which presents the clock uncertainty and network delay with two values: low end and high end. It takes an interval format, but it indicates a single point (point-interval-based timestamp). In contrast, we define durative events and give a new interval-based timestamp to the composite events based on the interval semantics (see Section 4). For primitive events, either a point-based or point-interval-based timestamp is applicable. Point-interval-based timestamp is an accuracy representation, and it is distinct from interval-based timestamp representing duration of events.

Recent progression of GPS technology makes us reconsider the use of GPS in distributed systems including wireless network environments in spite of GPS not being suitable for use in a large class of smart devices due to its high power consumption and the required line of sight to the GPS satellites. GPS may be the key for providing the accurate adjustment of the time at certain nodes where they are less resource constrained within the wireless ad hoc networks. The approach described in [15] shows that scalar clocks can eliminate vector clocks that have been used until now in distributed computing. GPS can solve time related problems, but because of the cost and condition of the GPS receipt, it may take some time to be deployed in the real world. We propose a coordinated approach with GPS and w/o GPS environments for time synchronization mechanism for wireless network environments. For w/o GPS scenario, we propose temporal ordering events using time synchronization mechanism [16]. For wired network environments where NTP and GPS can be deployed, we use interval-based time system described in [11]. Both approaches use interval-based timestamp representing inaccuracy of a single point time. Wireless ad hoc networks may be deployed with relay nodes to Internet backbones in many real world scenarios, it is possible relay nodes may have capability to GPS connection. Therefore transformed interval-based timestamps will be comparable to the timestamps created in wired network environments as far as event detection in wireless networks runs at nodes containing enough accuracy as GPS based time synchronization environments. Fig. 1 depicts that events from sensor networks are aggregated at node B with a transformed timestamps and passed through 802.11 network environments where GPS based time synchronization is deployed towards a subscriber node in Internet environments.

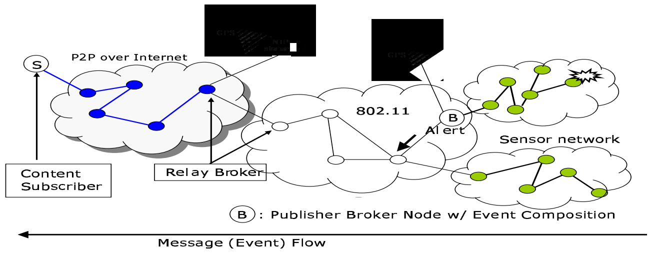


Figure 1: Message Propagation over Hybrid Networks

4 Event Correlation Semantics

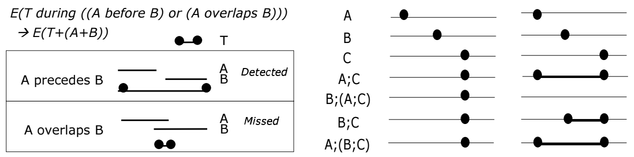
We define the composite events by expressions built from primitive events and the algebra operators. Operators of event algebra are defined informally in this section. We also support parameters and applying parameters helps to define unambiguous semantics of event detection and supports resource constrained environments. In wireless ad hoc networks, the event algebra must be restricted so that only a subset of of all possible occurrences of complex events will be detected. We provide basic operators which have the potential of easily expressing all semantics and capable of restricting expressions. Also an interval-semantics supports more sensitive interval relations among the events for the environments where more real-time concerns are c cal such as wireless networks or multi-media systems. The temporal operators introduced in [1] are not defined uniformly in many applications, and we define precise timing constraints (see Table 1).

In this paper, we focus on semantics of composite events, and methods to implement semantics (e.g., FSA, Petri nets, graphs) including higher language definition will be left for future work. We implemented a simple prototype based on a simple automata similar to described in [14] with support of parameterized values and time constraints.

Composite Event Operators: The event operators are defined informally as follows (see Table 1 for timing constraints).

- **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 during B, and A starts B* in Table 1.
- **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;NULL;B$_0$): denotes that there is event A and event B occur contiguously without any other events occurrence at the meeting time unit.
- **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).
- **Position A$^{[i]}$:** The position $A^{[i]}$ defines the occurrence of the i^{th} event A of a sequence of event set $A, i \in \mathbb{N}$.
- **Subset A$_S$:** Event A occurs if it is a subset defined in $s \in$ S. The subseting 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 Cambridge. Event A is valid only when subsetting 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.

Temporal Conditions: Defining temporal conditions for the semantics of composite events could be tricky, especially when timing constraint is important such as for processing transactions. This may cause an incorrect interpretation against the intuitive interpretation of the user. Fig.2(a) depicts a composite event E (*snow storm alert*): during the period when event A (*humidity stays up 60%*) occurs followed by event B (*wind blows towards south*), if event T (*temperature goes down below zero degree*) occurs. Two situations are shown. If we follow the interpretation of temporal conditions described in [6], in the first situation, event E is detected and in the second situation, event E is missed. In [6], *overlaps* and *during* only comprises the period when two events are simultaneously occurring, while every other operator 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 find consistent results.



(a) Semantics Ambiguity (b) Point and Interval

Figure 2: Semantic Ambiguity

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.2(b), 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, B followed by 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.

Event Context: Adding the policy defining the constraints provides the way to modify the operator semantics. This parameter-dependent algebra can accommodate different policies on event consumptions. 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 such as keeping most recent instance for future use. For example, for event consumption policy, three contexts *unrestricted*, *recent* and *chronicle* can be defined. Snoop [3] uses these contexts, but it is not capable to apply an individual context to different event operators.

Subset rule defines the subset of events to detect. Ideally the subset rule should interfere as little as possible

	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 during 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 in the same way 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:

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

where ρ is maximum clock skew.

Table 1: Interval Semantics - Timestamp for Composite Events

the unrestricted semantics. None of the removed instances should have a crucial impact on the detection of enclosing detection. For example, for non-duplicates policy, the subset rule can be restricting an event stream that does not contain multiple instances with the same end time.

5 Experiments

We show a brief result of experiments: time restricted composite event detection. Further experiments are in progress. Consider as an example, the event expression $A; B_5$: During the detection of this expression, all instances of event A that ended more than 5 time units ago, except the one with the latest start time, can be discarded. Fig. 3 shows the simulation of memory usage with the number of event states to be kept. For the detection policy, the most recent instance of A is used. Hundred random events are produced, and the time restriction value is set to 10, which may be relatively large number for the dynamic wireless environments. The time restriction is a similar concept of event detection with a sliding window, however with our approach, semantics are unambiguous and capability on individual definition for each event gives another advantage. The number of detected composite events is also shown in Fig. 3 that shows only half of composite events are detected when time restriction is specified. Whether undetected composite events imposes loss of information or reduction of overhead depends on the definition of composite events.

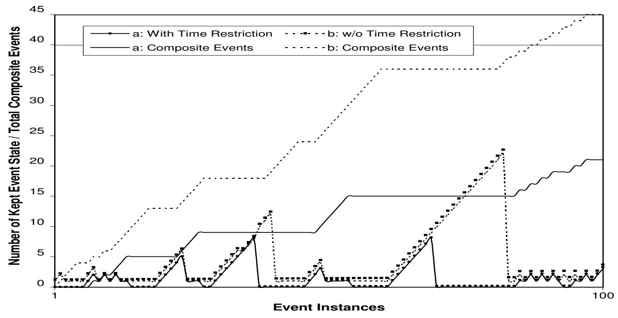


Figure 3: Time Restriction

6 Related Work

Many 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. Snoop [3] is an event specification language for active database, which informally defines event contexts. Transition from centralized system to distributed system brought the new challenge to deal with time. Snoop presents an event-based model for specifying timing constraints to be monitored and process both asynchronous and synchronous monitoring of real-time constraints. In [12], occurrence time of various event instances for time constraint specification is described. GEM [13] allows additional conditions, including timing constraints to combine with event operators for composite event specification. In event notification service, reduction of network traffic can be achieved by the event correlation [14]. The composite events in Cambridge Event Architecture [7] describes an object-oriented system with an event algebra that is implemented by nested push-down FSA to handle parameterized events. The event algebra developed in [7, 2] provides time restricted sequence and conjunction. See Section 3 for related work of time systems.

7 Conclusions and Future Work

In this paper, we analyze the composite event semantics, focusing on time related issues such as temporal ordering and interval-based semantics especially from the aspects of wireless network environments. We present event correlation semantics defining precise complex temporal relationships among correlated events using interval-based semantics. For real-time temporal event ordering, we coordinate GPS based and w/o GPS time synchronization environments using interval-based timestamp. Recent evolution of wireless networks makes the events flow from tiny sensor networks to Internet scale P2P systems among event broker grids. Our approach will provide event correlation in heterogeneous network environments. Work is ongoing on optimizing the complex event composition forms into an expression that can be more efficiently implemented in wireless ad hoc networks.

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] Carlson, J. et al. An Interval-Based Algebra for Restricted Event Detection. *Proc. FORMATS*, 2003.

[3] Chakravarthy, S. et al. Snoop: An expressive event specification language for active databases. *Data Knowledge Engineering*, 14(1), 1996.

[4] Elson, J. et al. Wireless Sensor Networks: A New Regime for Time Synchronization. *Hotnets-I*, 2002.

[5] Gatziu, S. et al. Detecting Composite Events in Active Database Systems Using Petri Nets. *RIDE-AIDS*, 1994.

[6] Gomez, R. et al. Durative Events in Active Databases. *Proc. ICEIS*, 2004.

[7] Hayton, R. OASIS: An Open architecture for Secure Interworking Services. *PhD thesis, Univ.of Cambridge*, 1996.

[8] Hinze, A. et al. A Meta-service for Event Notification. *Proc. CoopIS*, 2004.

[9] Kopetz, H. et al. Real-time systems development: The programming model of MARS. *Proc. ISADS*, 1993.

[10] Lamport, L. et al. Time, Clocks, and the Ordering of Events in a Distributed Systems. *CACM*, 21(4):558-565, 1978.

[11] Liebig, C. et al. Event Composition in Time-dependent Distributed Systems. *Proc. 4th CoopIS*, 1999.

[12] Liu, G. et al. A Unified Approach for Specifying Timing Constraints and Composite Events in Active Real-Time Database Systems. *Proc. IReal-TTA Symposium*, 1998.

[13] Mansouri-Samani, M. et al. GEM: A Generalized Event Monitoring Language for Distributed systems. *Distributed systems Engineering Journal*, 4(2), 1997.

[14] Pietzuch, P. et al. Composite Event Detection as a Generic Middleware Extension. *IEEE Network Magazine*, 2004.

[15] Prakash, R. et al. Causality and the Spatial-Temporal Ordering in Mobile Systems. *Mobile Networks and Applications*, 9(5):507-516, 2004.

[16] Roemer, K. Time Synchronization in Ad Hoc Networks. *MobiHoc01*, 2001.

[17] Schwiderski, S. Monitoring the Behavior of Distributed Systems. *PhD thesis, University of Cambridge*, 1996.