Object Tracking using Durative Events

Eiko Yoneki and Jean Bacon

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

Abstract. This paper presents a distributed middleware architecture based on a service-oriented approach, to manage high volume sensor events. Event management takes a multi-step operation from event sources to final subscribers, combining information collected by wireless devices into higher-level information or knowledge. An event correlation service provides sophisticated event filtering, aggregation and correlation over time and space in heterogeneous network environments. An experimental prototype in the simulation environment with real world data produced by the Active BAT system is shown.

1 Introduction and Background

The majority of current middleware for Wireless Sensor Networks (WSNs) is based on the data centric approach, and a fundamental idea naturally came from database management systems (DBMS). The database community has taken the view that declarative programming, through a query language, provides the right level of abstraction for accessing, filtering, and processing relational data. The middlewares that take a database approach such as [3] provides an interface for data collection but they do not provide general purpose distributed computation. For example, it is complex to implement arbitrary aggregation and filtering operators and communication patterns with query languages. Thus, more general interfaces for global network programming are desirable.

Recent evolution of ubiquitous computing with a dramatic increase of event monitoring capabilities by wireless devices and sensors requires an open platform for users to utilize seamlessly various resources in physically interacting environments, unlike the traditional closed network setting for specific applications. There has been an effort to architect middleware for such environments using service oriented architecture (e.g. RUNES [6] and P2PComp [1]). When designing the middleware for sensor networks, heterogeneity of information over global distributed systems must be considered. The sensed information by the devices is aggregated and combined into higher-level information or knowledge.

Service Oriented Architecture (SOA) is a well proven concept for distributed computing environments. It decomposes applications, data, and middleware into reusable services that can be flexibly combined in a loosely coupled manner. SOA maintains agents that act as software services performing well-defined operations. This paradigm enables the users to be concerned only with the operational description of the service. All services have a network addressable interface and communication via standard protocols and data formats (i.e., messages). SOA can deal with aspects of heterogeneity, mobility and adaptation, and offers seamless integration of wired and wireless environments.

Generic service elements are context model, trust and privacy, mobile data management, configuration, service discovery, event notification, and the following are the key issues addressed for our design.

- Flexible discovery mechanisms for ad hoc networks, which provide the reliable discovery of newly or sporadically available services.
- Support for adaptive communication modes, which provides an abstract communication model underlying different transport protocols. Notably, event-based communication is suitable for asynchronous communication.

Peer-to-peer networks and grids offer promising paradigms for developing efficient distributed systems and applications. Grids are essentially P2P systems. The grid community recently initiated a development effort to align grid technologies with Web Services: the Open Grid Services Architecture (OGSA) [4] lets developers integrate services and resources across distributed, heterogeneous, dynamic environments and communities. The OGSA model adopts the Web Services Description Language (WSDL) to define the concept of a grid service using principles and technologies from both the grid and Web Services. The architecture defines standard mechanisms for creating, naming, and discovering persistent and transient grid-service instances. The convergence of P2P and Grid computing is a natural outcome of the recent evolution of distributed systems, because many of the challenging standards issues are quite closely related.

The Open Services Gateway Initiative (OSGi) [5] is focused on the application layer and open to almost any protocol, transport or device layers. The three key aspects of the OSGi mission are multiple services, wide area networks, and local networks and devices. Key benefits of the OSGi are that it is platform independent and application independent. In other words, the OSGi specifies an open, independent technology, which can link diverse devices in the local home network. The central component of the OSGi effort is the services gateway. The services gateway enables, consolidates, and manages voice, data, Internet, and multimedia communications to and from the home, office and other locations.

We propose a distributed middleware architecture that envisages an integrated service oriented architecture to manage high volume sensor events in global computing. The current mainstream deployment of sensor networks collects all the data from the sensor networks and stores them in the database and data analysis is preceded from there. The proposed architecture deploys distributed gateways to collaborate data management over hybrid network environments. The publish/subscribe paradigm is used for asynchronous communication, performing data aggregation and distributing filtered data to other networks based on contents. We simulate distributed gateways with the real world data produced by the Active BAT system [2].

This paper continues as follows: section 2 describes the middleware architecture, section 3 briefly recalls the durative event model defined in our previous work [7], section 4 reports an experimental prototype, section 5 describes related works and it concludes with section 6.

2 Middleware Architecture

We have developed a generic reference architecture applicable to any ubiquitous computing space. The middleware contains separate physical, sensor components, event broker, service, and service management layers including an open application interface. An implementation of a reference architecture is in progress.

A service is an interesting concept to be applied in WSNs. It may be a role on a sensor node, or a function providing location information. Services allow cascading without previous knowledge of each other, and enable the solution of complex tasks, where functional blocks are separated in order to increase flexibility and enhance scalability of sensor network node functions. A key issue is to separate the software from the underlying hardware and to divide the software into functional blocks with a proper size and functionality. Another important issue is that the sensed data should be filtered, correlated, and managed at the right time and place when they flow over heterogeneous network environments. It is not easy to provide reliable and useful data among the massive information from WSNs. An event correlation based on our previous work [7] is integrated with service composition.

2.1 Service Semantics

Service semantics is an important issue, in addition to the service definition, so that services can be coordinated in the space. The model of the real world is of a collection of objects, where objects maintain state using sensor data, and applications' queries and subscriptions are a relevant sets of objects. Fig.1 shows an example of object mappings among applications, middleware and sensor components. Objects are tightly linked to event types in an event broker. Exploiting semantics will let the pervasive space's functionality and behaviour develop and evolve. Space specific ontologies will enable such exploitation of knowledge and semantics in ubiquitous computing.

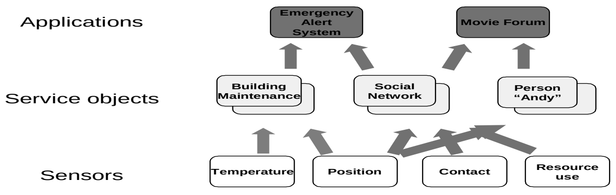


Fig. 1: Mapping Real World to Applications

2.2 Layer Functionality

Fig.2 depicts the overview of the middleware, and the brief functionality of each layer is shown below.

Physical Layer: This layer consists of the various sensors and actuators.
Sensor Component Layer: A sensor component layer can communicate with a wide variety of devices, sensors, actuators, and gateways and represent them to the rest of the middleware in a uniform way. A sensor component effectively converts any sensor or actuator in the physical layer to a software service that can be programmed or composed into other services. Decoupling sensors and

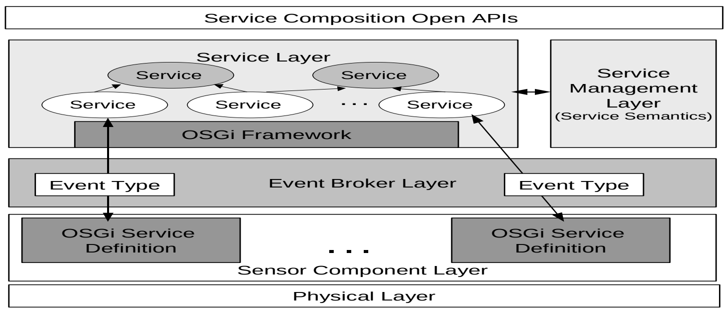


Fig. 2: Middleware Architecture with Wireless Sensor Data

actuators from sensor platforms ensures openness and makes it possible to introduce new technology as it becomes available.
Event Broker Layer: This layer is a communication layer between Sensor components and the Service layer. It supports asynchronous communication using the publish/subscribe paradigm. Event filtering, aggregation, and the correlation service is a part of this layer.
Service Layer: This layer contains the Open Services Gateway Initiative (OSGi) framework, which maintains leases of activated services. Basic services represent the physical world through sensor platforms, which store service bundle definitions for any sensor or actuator represented in the OSGi framework. A sensor component registers itself with the service layer by sending its OSGi service definition. Application developers create composite services via the Service Management Layer's functions to search existing services and using other services to compose new OSGi services. Canned services, which may be useful globally, could create a standard library.
Service Management Layer: This layer contains an ontology of the various services offered, and the appliances and devices connected to the system. Service advertisement and discovery use service definitions and semantics to register or discover a service. Service definitions are tightly related to the event types used for communication in the Event Broker Layer including composite formats. The reasoning engine determines whether certain composite services are available.
Application Interface: An application interface provides open interfaces for applications to manage services, including the management of contexts.

3 Durative Events and Interval-based Semantics

In [7], we defined a unified semantics, combining traditional event composition and data aggregation in WSNs. For event detection, we introduced a parameterized algebra. Parameters include time, selection, consumption, precision, and subset policies. This approach defines unambiguous semantics of event detection and supports resource constrained environments. The semantics is integrated with the service composition engine in the middleware described in Section 2.

In our event model, a primitive event is the occurrence of a state transition at a certain point in time. Each event has a timestamp associated with the occurrence time. The timestamp is an approximation of the event occurrence time. In most event algebras, each event occurrence, including composite events, is associated with a single value indicating the occurrence time. This may result in unintended semantics for some operator combinations, for example nested sequence operators. We define a composite event with duration and give a new interval-based timestamp to a composite event based on an interval semantics. 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. An event can have a space stamp indicating certain location, relative location, and grouping (e.g. position (x,y,z), global id). Complex timing constraints among correlated event instances are precisely defined (see Appendix in [7]).

Composite events are defined by expressions built from primitive and composite events and algebraic operators. Operators consist of *Conjunction, Disjunction, Concatenation, Sequence, Concurrency, Iteration, Negation, Selection, Spatial Restriction, and Temporal Restriction* (See [7] for the detail). We also support parameters, which help to define unambiguous semantics of event detection and support resource constrained environments. 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. The following example illustrates the use of the operators to describe composite events.
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.

4 Prototype with the Active BAT System

Sentient computing is a type of ubiquitous computing which uses sensors to perceive its environment and react accordingly. A use of the sensors is to construct a world model, which allows location-aware or context-aware applications to be constructed. One research prototype of a sentient computing system was the work at AT&T Laboratories in the 1990s and the research continues at our Computer Laboratory by means of the Active BAT system [2]. This is a low power, wireless, indoor location system accurate up to 3cm. It uses an ultrasound time-of-light trilateration technique to provide accurate physical positioning.

Users and objects carry Active BAT tags. In response to a request that the controller sends via short-range radio, a BAT emits an ultrasonic pulse to a grid of ceiling mounted receivers. At the same time that the controller sends the radio frequency request packet, it also sends a synchronized reset signal to the ceiling sensors using a wired serial network. Each ceiling sensor measures the time interval from reset to ultrasonic pulse arrival and computes its distance from the BAT. The local controller then forwards the distance measurements to a central controller, which performs the trilateration computation. Statistical pruning eliminates erroneous sensor measurements caused by a ceiling sensor hearing a reflected ultrasound pulse instead of one that travelled along the direct path from the BAT to the sensor. The SPIRIT (SPatially Indexed Resource Identification

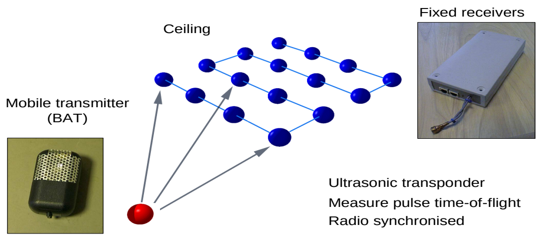


Fig. 3: Active BAT

and Tracking) [2] provides a platform for maintaining spatial context based on raw location information derived from the Active BAT location system. It uses CORBA to access information and spatial indexing to deliver high-level events such as 'Alice has entered the kitchen' to listening context aware applications. SPIRIT models the physical world in a bottom up manner, translating absolute location events for objects into relative location events, associating a set of spaces with a given object and calculating containment and overlap relationships among such spaces, by means of a scalable spatial indexing algorithm. However, this bottom-up approach is not as powerful in expressing contextual situations.

4.1 Distributed Gateways

The current Active BAT system employs a centralized architecture, and all the data are gathered in the database, where computational power is cheap. The Active BAT system, as described, is expensive to implement in that it requires large installations, has a centralized structure. The centralized structure allows for easy computation and implementation, since all distance estimates can be quickly shipped to a place where computational power is cheap. Moreover, the active mobile architecture facilitates the collection of multiple simultaneous distance samples at the fixed nodes, which can produce more accurate position estimates relative to a passive mobile architecture.

It is inherently scalable both in terms of sensor data acquisition and management as well as software components. However, when constructing real-time mobile ad hoc communications with resource-constrained devices, a distributed coordination must be supported, so that mobile device users can subscribe certain information promptly. We simulate each room and corridors hold gateway nodes (see the location map Fig.4), which is capable to participate in event broker grids. The software design follows the service-oriented architecture described in Section 2. Thus, each local gateway node performs event filtering and correlation. Each local node registers the service that associates states with abstractions such as 'Andy in the room SN04'. These states are decomposed to the units

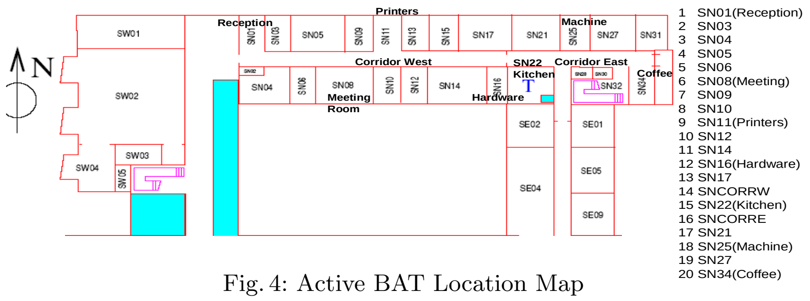
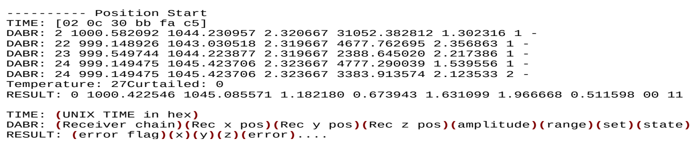


Fig. 4: Active BAT Location Map

operable by the event broker grid, where event correlation and aggregation and filtering are supported. The details of high-level language for service composition substantially event type definition is still under development, and out of scope of this paper. The used data is taken on May 20th in 2003 for 24 hours. The total number of original events received by the ceiling units is around 400,000, and a sample is shown in Fig.5. On average a sighting probably gets around 10 receptions, of which perhaps 2 will be 100% noise, 2 or so will be too noisy and will be rejected and the rest will be used for the final estimate.The format is:



```
---------- Position Start
TIME: [02 0c 30 bb fa c5]
DABR: 2 1000.582092 1044.230957 2.320667 31052.382812 1.302316 1 -
DABR: 22 999.148926 1043.030518 2.319667 4677.762695 2.356863 1 -
DABR: 23 999.549744 1044.223877 2.319667 2388.645020 2.217386 1 -
DABR: 24 999.149475 1045.423706 2.323667 4777.290039 1.539556 1 -
DABR: 24 999.149475 1045.423706 2.323667 3383.913574 2.123533 2 -
Temperature: 27Curtailed: 0
RESULT: 0 1000.422546 1045.085571 1.182180 0.673943 1.631099 1.966668 0.511598 00 11

TIME: (UNIX TIME in hex)
DABR: (Receiver chain)(Rec x pos)(Rec y pos)(Rec z pos)(amplitude)(range)(set)(state)
RESULT: (error flag)(x)(y)(z)(error)....
```

Fig. 5: Active BAT Raw Events

The 'set' value can be 1 or 2 and represents whether the pulse was the first received or the second (so the pulses marked 2 are irrelevant to positioning, but there for other uses). A '1' pulse can be assigned a state A (accepted and used in the positioning calculation) or R (rejected and not used). After the position calculation, the total number of events around 200,000 are created (see Fig.6). This shows BAT data after the location of the user is calculated, which consists of timestamp, user, area, coordination (X, Y, Z) and orientation.

```
30408.802618,10,SN09,1002.335327,1033.320801,1.261441,-22.443605
30408.856115,10,SN09,1002.520386,1033.289429,1.251856,-20.335649
30409.063099,10,SN09,1002.533203,1033.279297,1.285185,-20.326197
30409.112594,10,SN09,1002.732910,1033.234863,1.270585,-22.712467
30409.315079,10,SN09,1002.921448,1033.175903,1.271525,-54.598316
30409.370575,10,SN09,1002.994690,1033.126587,1.283121,-56.499645
30409.564561,10,SN09,1003.170227,1033.044556,1.298443,-52.581676
```

Fig. 6: Active BAT Location Events

4.2 Experiments

We performed several event correlations, and among those, we show the use of durable events below. Fig.7 depicts the number of events over the local gateway nodes without durable event and Fig.8 shows the same operation with durable event compositions. During this experiment, 21 BAT holders participated. The result shows dramatic reduction of event occurrences by the use of durable events, where the state of the target object is maintained.

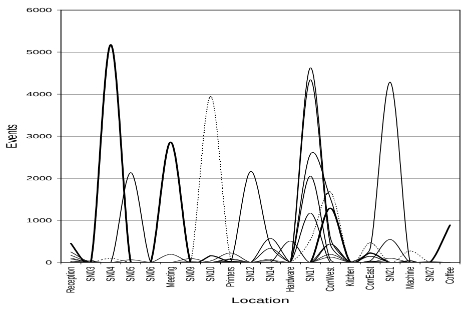


Fig. 7: Events over Locations

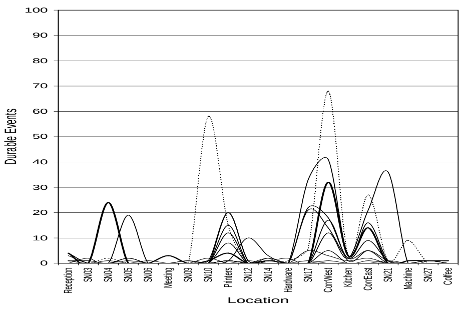


Fig. 8: Durable Events over Locations

Fig.9 and Fig.10 depict the events identified on the BAT holders Andy and Brian. Andy's office is most likely the location 3 (room SN04), where the highest number of events is recorded. Brian's office is the location 8 (room SN10), where also the large number of events is produced. See the numbers corresponding to the location is described in Fig.4. Fig.9 and Fig.10 show the events over the location, however they do not indicate when they occurred.

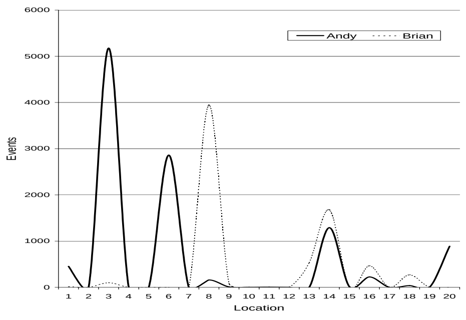


Fig. 9: Events over Locations

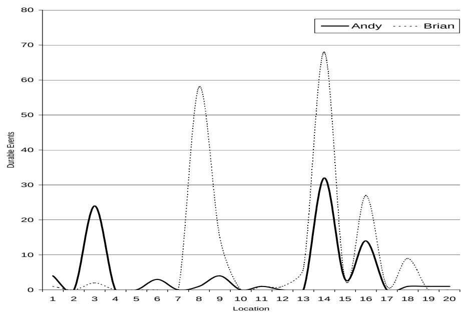


Fig. 10: Durable Events over Location

Fig.11 and Fig.12 depict the events over the timeline (24 hours). Most activities are recorded during the day time. Durable events composition over the timeline (24 hours) shows significant reduction of the number of events.

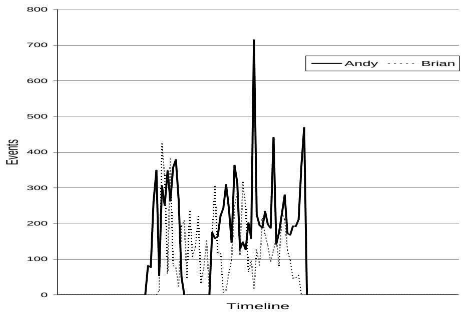


Fig. 11: Events over Time

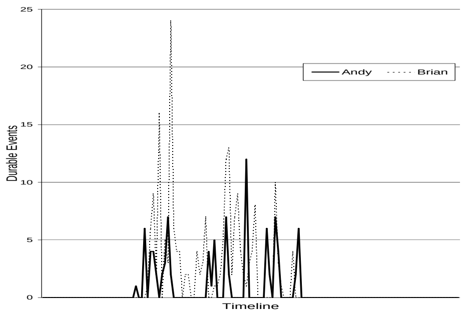


Fig. 12: Durable Events over Time

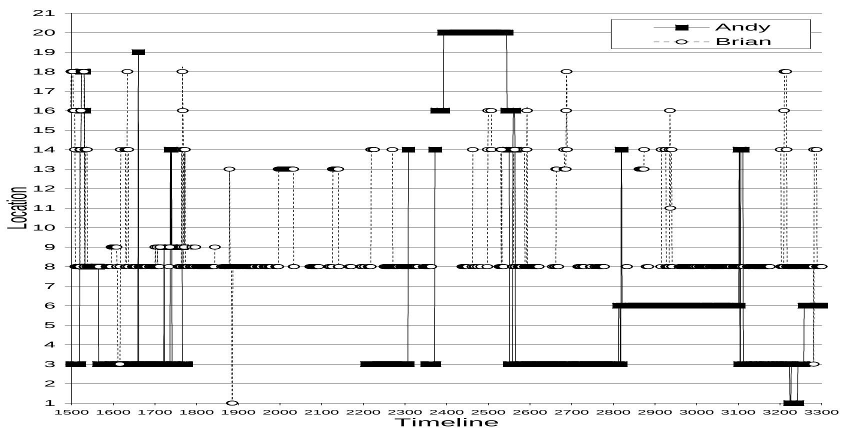


Fig. 13: Events over Location and Time

Fig.13 traces Andy and Brian over the time and location between time unit 1500 and 3300. One unit is 15 seconds, and 7 hours and half duration of activities is shown. It looks like Andy and Brian spent a lot of time in the location 8 (room SN10), where Brian's office as well at the both corridors.

Fig.14 shows the specific period, when they were positioned at the corridor west. The composite event *(Brian;Andy)corrwest* is detected at time unit 2564.

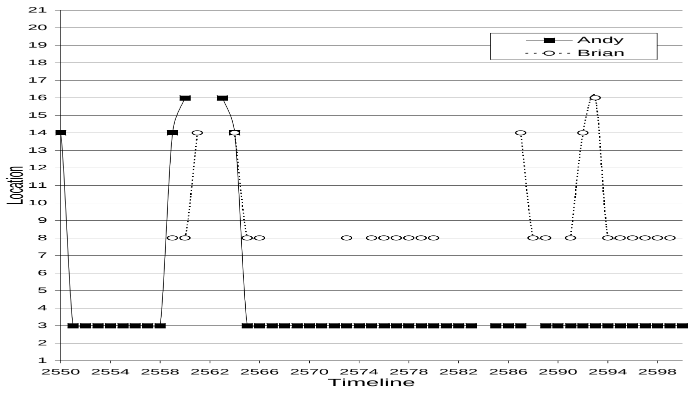


Fig. 14: Composite Event - (Brian;Andy)corrwest

In Fig.15, the detection of composite event *(Andy+Brian)SN25(machine)* is shown at the time unit between 1523 and 1529. A local gateway can detect this correlation, if the composite event is subscribed through the event broker. This composition could be a part of services provided by the service grid.

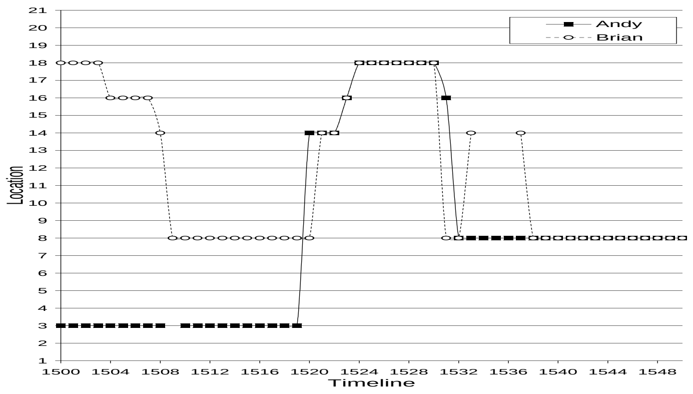


Fig. 15: Composite Event - (Andy+Brian)machine

4.3 Temporal Ordering in Active BAT System

The applications derived from Active BAT have high accuracy and real-time tracking requirements. Problems of time synchronization and coordination amongst beacons are easily resolved, because these systems are wired and have a centralized controller. The timestamp is derived from a Global Clock Generator (GCG), which is a hardware clock that sends 'ticks' to every component of the system over a serial link. When a location is computed, the location message is timestamped using the GCG. In general, GCG delay is in the order of microseconds, and the slowest part of the system is the bit that waits for ultrasound to propagate (speed of sound) after a position is requested but before a position can be calculated. This delay is used to measure the distance between the BAT and the receiver at the ceiling. Once the location is calculated, the message then has to travel up to SPIRIT (order of milliseconds), and the event will be generated. However, no reliable information on that delay is considered. However, when gateways are distributed temporal ordering of events requires more complex time synchronization. The implementation of temporal ordering mechanism described in [7] is in progress. The current experiment assumes that all timestamps are properly synchronized.

5 Conclusions and Future Works

For WSN applications, distributed programming abstractions and middleware will be key technologies. In this paper, we introduce a middleware architecture and present an experimental prototype of object tracking with real world data produced by the Active BAT system. The tracking system uses the proposed middleware, which provides an event correlation service. Our integrated approach of event correlation, filtering and aggregation can express the complex composite event for tracking conditions. 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. Event broker grids should seamlessly disseminate relevant information generated by deeply embedded controllers to all interested entities in the global network, regardless of specific network characteristics, leading to a solution to construct large distributed systems over dynamic hybrid network environments. We are working on a complete implementation including various timestamping environments and different communication protocols.

Acknowledgment. This research is funded by EPSRC (Engineering and Physical Sciences Research Council) under grant GR/557303. We would like to thank Robert Harle (Digital Technology Group in University of Cambridge) for providing the Active BAT data.

References

1. Ferscha, A. et al. A Light-Weight Component Model for Peer-to-Peer Applications. *Proc. MDC*, 2004.
2. Harter, A. et al. The Anatomy of a Context-Aware Application. *Proc. MobiCom*, 1999.
3. Madden, S. et al. TAG: A tiny aggregation service for ad-hoc sensor networks. *Proc. of Operating Systems Design and Implementation*, 2002.
4. OGSA Working Group, http://www.ggf.org/ogsa-wg/.
5. OSGi, http://www.osgi.org.
6. Reconfigurable Ubiquitous Networked Embedded Systems . *http://www.ist-runes.org*, 2005.
7. Yoneki, E. and Bacon, J. Unified Semantics for Event Correlation over Time and Space in Hybrid Network Environments. *Proc. CoopIS*, 2005.
8. Zhang, W. et al. Optimizing Tree Reconfiguration for Mobile Target Tracking in Sensor Networks. *Proc. Infocom*, 2004.