Providing Hard Real-Time Guarantees in Context-Aware Applications: Challenges and Requirements

Malika Boulkenafed, Barbara Hughes, René Meier, Gregory Biegel, Vinny Cahill
Distributed Systems Group,
Department of Computer Science,
Trinity College Dublin, Ireland.
Email: {first.last}@cs.tcd.ie

ABSTRACT

Context-aware applications rely on the ability to perceive the state of the surrounding environment. In this paper, we address a class of such applications where real-time guarantees are required on top of mobile ad hoc networks. While guaranteed timeliness is a critical requirement, the unpredictability of dynamic wireless networks adversely impacts such guarantees. Therefore, we identify the challenges and the requirements on different architectural levels in order to provide timeliness guarantees. None of the existing systems have succeeded in providing adequate solutions to all of the identified requirements. Therefore, we describe a cross-layer architecture that supports the development of real-time context-aware applications for wireless networks, in particular, ad hoc networks.

This cross-layer architecture is based on three main components. (i) Sentient objects - mobile intelligent software agents that extract, interpret and use context information. (ii) Event-based real-time middleware supports communication between sentient objects and provides hard real-time guarantees within adaptable geographic spaces. (iii) A real-time routing and resource reservation protocol attempts to discover and maintain real-time constrained routes within these proximities in a multi-hop ad hoc network.

I. INTRODUCTION

Some 30 years of research have gone into creating distributed computing systems, and nearly 20 years have been invested in experiments with mobile computing [1]. This background and today's developments in miniaturization and wireless operation have given rise to an important new class of mobile context-aware applications. Context-aware applications are a large and important subset of the overall set of ubiquitous computing applications. *Context* is the commonly accepted term used to describe the state of the environment in which an application operates. Context is potentially made up of a number of attributes describing location, identity, activity, bandwidth, power and a host of other parameters. A mobile application is called *context-aware* if it has the ability to extract, interpret and use such information obtained from its environment. These applications will typically run on communication-enabled heterogeneous mobile devices and be capable of interacting with each other and with their surrounding environment via a range of sensors and actuators.

Such context-aware applications interact with the physical environment and therefore inherently exhibit timeliness requirements. The correctness of context information sensed from a diverse range of sensors not only depends on their logical correctness, but also on the time at which the result is produced and disseminated. Therefore, we can categorise sensed information based on its processing/propagation timeliness requirements. We identify three types of timeliness requirements: hard real-time, soft real-time and time-free. If the processing or the propagation of sensed information requires hard real-time guarantees, this means that there is an upper bound on the time to process this information as well as timeliness constraints on its propagation. Providing such guarantees implies time-bounded medium-access and routing latency. When the processing or the propagation of sensed information requires soft real-time guarantees, then timeliness constraints may be violated under load and fault conditions without critical consequences. Finally, if the processing or the propagation of sensed information is time-free, this means that no upper time bound is known for a process to execute a

computation or to send a message.

Mobile context-aware applications in the ubiquitous computing domain are challenged by the dynamism and flexibility of today's wireless networks, especially with the advent of protocols supporting multi-hop ad hoc communication. To take advantage of such wireless networks, mobile context-aware applications have to be adaptive, and capable of dealing with network topology changes in a flexible and unsupervised way. While guaranteed timeliness is a critical requirement for many mobile context-aware applications, the unpredictability of topology changes in dynamic wireless networks and their resource constraints adversely impact the guarantees available for timeliness properties. Achieving hard real-time guarantees in a resource-constrained, wireless ad hoc network is potentially impossible without restricting the characteristics of real-time applications, the environment or both. Restricting the computing environment of hard real-time applications by assumption reduces the severity of the challenges to be overcome, but may also reduce the general applicability of the approach.

Therefore, we identify the main challenges and the requirements on different architectural levels in order to provide such guarantees. On the application level, the main challenge consists in providing an adequate programming model to support the developers of context-aware applications. On the middleware level, the asynchronous communication inherent in the nature of context-aware applications' interaction with their external environment is best captured in a generative, event-based communication model [2]. However, the challenge remains to extend event-based communication to provide hard real-time guarantees while supporting ad hoc wireless networks. Finally, the greatest challenge at the routing level consists in accomodating the unpredictability and the dynamics of wireless ad hoc networks with timeliness constraints for guaranteed hard real-time.

Our proposal exploits the rationale observed in [3], i.e., the relevance of context to a particular geographical area, to guarantee real-time constraints within specific proximity-bounds only. Thus, we solve the problem by reducing the area of the environment where real-time communication is guaranteed to within the defined proximity bounds only. However, the dynamics of the environment impact the real-time guarantees available even within a bounded proximity. In this case, we dynamically adapt the proximity bound to maintain the required real-time guarantees. This dynamic proximity, or space-elastic, adaptation is at the core of our solution tackling the challenge of hard real-time communication in a mobile environment.

This paper describes our proposal through a cross-layer architecture which builds on three main components. *Sentient objects* - mobile intelligent software agents that extract, interpret and use context information to drive their behaviour. *Event-based real-time middleware* supports communication between sentient objects and provides hard real-time guarantees within adaptable geographic spaces. A *real-time routing and resource reservation* protocol discovers and maintains real-time constrained routes within adaptable geographic proximities by using a wireless time bounded medium access protocol in a multi-hop ad hoc network. Our cross-layer architecture supports the application developer in the following key ways:

- Through the sentient object model, it provides: (i) abstractions for sensors and actuators, thus relieving the developer of the burden of low level interaction with various hardware devices; (ii) a probabilistic mechanism for fusing multi-modal fragments of sensor data together in order to derive higher-level context information; (iii) an efficient approach to intelligent reasoning based on a hierarchy of contexts; (iv) a simple API to specify producers and consumers of events.
- Through the middleware, it provides an event-based communication mechanism for interaction between sensors, sentient objects and actuators.
- Through the space-elastic model, it provides: (i) timeliness guarantees within specified proximities; (ii) space-elastic adaptation and timely adaptation notification to the originating event producer.

The first contribution of this paper consists in a clear identification of the main challenges and the requirements of context-aware applications that exhibit hard real-time guarantees in wireless ad hoc networks. The second contribution of this paper is the definition of a cross-layer architecture that supports the development of hard real-time context-aware applications for wireless networks, in particular, ad hoc networks.

The remainder of this paper is structured as follows. Section II describes the challenges and the requirements encountered when attempting to provide hard real-time guarantees. Section III discusses some of the related work highlighting proposals to deal with these challenges and requirements. However, none of the existing systems has succeeded in providing an adequate solution to all the challenges and the requirements. Therefore, in Section IV we describe our solu-

tion to provide context-aware applications with hard real-time guarantees. Finally, we conclude this paper by overviewing, in Section V, our contributions and discussing our current and future work.

II. Challenges and Requirements

The main feature of context-aware applications is their ability to perceive the state of the surrounding environment. Components of these applications are expected to be capable of acting in a decentralised fashion, based on the acquired context information and on their own knowledge. Such applications may be composed of many millions of interacting hardware and software components which may be scattered over buildings, cities, countries and continents, thus raising scalability concerns. Finally, these applications will have to cope with changing conditions in the underlying mobile environment. Therefore, such applications must be designed to evolve, while the underlying middleware support must be designed to be adaptable.

This section describes the challenges and the requirements encountered in providing real-time and in particular hard real-time guarantees for context-aware applications in wireless ad hoc networks. We identify these challenges and requirements at three levels as follows.

A. Application Level

One of the greatest challenges in developing context-aware applications, and one that has not yet been adequately addressed, is the provision of an adequate programming model to support the application developer. The major problem lies in providing a generic programming model promoting scalability, extensibility, and reuse of application components, and most importantly, supporting the acquisition, the fusion and the interpretation of context information gleaned from unreliable multi-modal sensors. At present, programmers are often required to write large amounts of code and interact with sensor and actuator devices at a low level in order to develop relatively simple applications.

In this section we discuss the major challenges to developing context-aware applications in mobile environments that must be addressed by a programming model.

- **Capture of context data.** In order to be able to interact with the physical environment, context-aware applications glean data from (potentially unreliable) multi-modal sensors. In addition to identifying the relevant sources of context data for a particular application, the application developer often has to write low-level code to interact with sensor hardware at the device protocol level. Such development is time-consuming, error-prone, and only accessible to fairly experienced programmers.
- **Uncertainty of context data.** Measurements made of the real world by sensors based on physical transducers will always contain a degree of *uncertainty* and *incompleteness*, which together result in an inherent unreliability of context data based on such measurements.
- **Representation of context data.** Context information needs to be organized in order to facilitate its interpretation. The selected representation format should be efficient to process and reason about by the application, in order to efficiently make useful inferences from this data.
- **Interpretation of context data.** Context-awareness is predicated on the accurate extraction, combination, and interpretation of context data. The interpretation may use an inference engine based on conditional rules coupled with a knowledge base.
- **Privacy.** Traditional concerns regarding privacy are amplified in context-aware applications which are predicated on access to a wide range of potentially sensitive data. If context-aware computing is to be embraced as a mainstream technology, privacy of sensitive data has to be assured, and application developers require appropriate tools to manage privacy and security.
- **Scalability.** Context-aware applications will form a part of an overall pervasive computing infrastructure consisting of very large and dynamic distributed populations of entities, and thus scalability of communication is an important consideration to application developers.
- **Synchrony.** Context-aware applications need to be notified asynchronously when new context data is available. Synchronous operations in context-aware applications imply expensive polling behaviour in order to determine when the requisite information is available. Such expensive communication behaviour is not suited to the resource constraints inherent in mobile devices.
- **Extensibility and reusability.** Support for extensibility and reusability is an essential and a challenging requirement of a programming model for context-aware applications. It is likely that in the future multiple, unanticipated types and sources of context will become available, and new applications will emerge that use existing sources of context. The ability to seamlessly integrate new sources of context data into applications,

and to re-use existing functionality is a key feature to the realisation of pervasive computing.

B. Middleware Level

In this section, we discuss the major challenges encountered at the middleware level when attempting to provide real-time guarantees.

- **Synchrony.** The loose coupling that characterizes applications in a wireless ad hoc network is inherent to dynamically changing populations of interacting entities and the dynamic reconfiguration of the connections between them. Therefore, appropriate middleware-level communication should accomodate such characteristics.
- **Group communication.** Group communication takes care of assembling mobile nodes that together allow to meet target functional and non-functional properties, and of further making failures due to the mobility of nodes transparent. The challenge then lies in the adaptation of the group communication service to the dynamic topology of the network.
- **Collaboration.** Context-aware applications are likely to involve ad hoc collaboration between mobile entities. The challenge consists in guaranteeing collaboration between mobile entities while meeting the QoS constraints.
- **Adaptability.** Middleware platforms must adapt their functionalities so as to best cope with possible resource constraints (energy, bandwidth) of mobile terminals as well as with the various QoS available. Furthermore, providing adaptation feedback to the application as additional context information allows it to undertake appropriate actions in order to adapt its behaviour.
- **Scalability.** The middleware platform should not need to change when the scale of the system increases. Rather, as the demand for a resource grows, it should be possible to extend the system to meet it. This issue is particularly challenging when attempting to provide hard real-time guarantees.
- **Resource preservation.** The middleware platform must minimize resource consumption on mobile nodes, and in particular energy, requiring minimizing message exchanges. The challenge consists in making adequate tradeoffs among messages exchange required for guaranteed timeliness, and resource consumption on mobile devices.
- **QoS.** Various QoS attributes may be considered. In particular, the following attributes appear to be the most dominant in the context of ad hoc networks: timeliness, reliability, security, performance and transactional behaviour.
- **Consistency and fault tolerance.** Given the highly dynamic nature of ad hoc networks, providing a consistent view of the network is very challenging. However, this may be considered as important context information and may be required by some applications. Furthermore, providing fault-tolerance requires a drastic tradeoff between replication and resource consumption on mobile devices.
- **Extensibility and reusability.** It is likely that in the future context-aware applications will exhibit new functional and non-functional requirements. The ability to seamlessly integrate new components into the middleware, and to re-use existing ones is a key feature for the realisation of context-aware computing.

The challenge remains to extend real-time communication middleware to provide hard real-time guarantees for dynamic wireless ad hoc networks, since existing real-time communication middleware provides only soft real-time guarantees and often makes the assumption that application components are stationary or that a fixed network infrastructure exists to facilitate communication.

C. Networking and Routing Level

In this section, we discuss the major challenges encountered at the networking and routing level when attempting to provide real-time guarantees.

- **Mobility and dynamic connectivity.** Ad hoc wireless networks are the most extreme example of dynamism in wireless networks with the possible mobility of all hosts in the network. In ad hoc networks a host is limited by the transmission range of its wireless interface. If a host wishes to communicate beyond its transmission range, then one or more of the hosts within its communication range would need to act as a router on behalf of this node. As nodes move in and out of range of other nodes, the connectivity and network topology changes dynamically.
- **Limited bandwidth availability.** Another challenging issue is introduced by the limitation in bandwidth availability. The size and the volume of real-time communication may be limited by this constraint.
- **Unpredictable latency.** The unpredictable latency for route determination and medium access (encountered at each hop) makes the estimation of end-to-end delivery latency very difficult and with a high probability of being inaccurate. Decisions based on inaccurate information

have unpredictable consequences that may be critical for real-time communication.

- **Intermittent connectivity.** The received signal strength (RSS) is continually changing due to the movement of the communicating and intermediary nodes. The RSS is also significantly affected by the geographical configuration and the transmission power of the wireless device. The changes in RSS lead to highly unpredictable connections between mobile nodes and increase link failure. The rate of link failure due to node mobility and changing RSS is one of the primary obstacles to providing hard real-time guarantees in ad hoc networks.
- **Adaptability.** The topology changes introduced by node mobility and wireless link failures must somehow be communicated to other nodes and to the application level in order for them to adapt their behaviour. Since communication and computation resources are limited, any communication overhead must be kept to a minimum.
- **Scalability.** Multi-hop ad hoc networks form a part of very large and dynamic distributed populations of entities, and thus scalability of communication is an important consideration. However, it may negatively impact the available real-time guarantees and may require to bound the size of the network to provide such guarantees.
- **Heterogeneity.** Ad hoc wireless networks are likely to be formed by heterogeneous lightweight computing devices, e.g., PDAs, third generation mobile phones, hand-held computers, or motes. These devices have different resource availability and require adapted protocols to avoid unnecessary overloading of their resource consumption.
- **Resource poverty.** Since the available bandwidth is limited and some wireless devices have severe energy constraints, relying for example on battery power. In addition, transmission range decreases as battery power reduces, causing significant weakening of the RSS. Hence, communication is an expensive operation in mobile ad hoc wireless networks in terms of bandwidth and energy consumption and therefore any overhead due to resource reservation, routing and scheduling must be kept to a minimum.
- **Providing end-to-end QoS.** Predictive routing protocols combining mobility prediction with partition anticipation and resource reservation are critical to provide end-to-end QoS in ad hoc wireless networks.

The time-varying capacity of wireless links, limited resources and node mobility make maintaining accurate routing information very challenging, if not impossible, in ad hoc wireless networks. Routing for real-time communication must ensure resource availability (e.g. bandwidth) whilst maintaining minimum latency. Routing decisions may be compromised by inaccurate network information and time-bounded route determination, where optimal routes may not be found within the time available.

III. Related Work

Whilst extensive support exists for addressing individual challenges such as the capture of context data [4], [5], the management of uncertain context data [6], [7], the efficient representation of context data [8], [9], and the management of privacy and trust [10], [11], there is no unifying model providing this support to the programmer in an easily accessible manner to enable the development of context-aware applications.

The inherent loose coupling that characterizes applications in a wireless ad hoc network promotes event-based communication as a natural design abstraction [2]. The real-time event-based communication paradigm has been recognized as an appropriate middleware-level communication scheme to connect autonomous components in large scale context-aware applications. However, existing event-based middleware only provide soft real-time guarantees. Existing research on event-based middleware for wireless networks has mainly focused on what may be termed nomadic applications. These applications are characterized by the fact that mobile nodes make use of the wireless network primarily to connect to a fixed network infrastructure, such as the Internet, but may suffer periods of disconnection while moving between points of connectivity. In contrast, context-aware applications are composed of mobile nodes that use the wireless network to communicate with each other within some common geographical area. Although these applications may use infrastructure networks, they will often use ad hoc networks to support communication without the need for an existing infrastructure. The other assumption made by existing event-based (real-time or not) middleware concerns the presence of centralised components to disseminate events which is not suitable for dynamic environments resulting from ad hoc networks. For instance, the event models of the CORBA Event Service [12], the CORBA Notification Service [13], TAO (The Ace Orb) RT Event Service [14], CONCHA (CONference system based on Java and CORBA Event Service CHAnnels) [15], and CEA (Cambridge Event Architecture) [16] are similar in that they all exploit mediator components through which event data is disseminated. In other

event models such as JEDI [17], SIENA [18] and ELVIN [19] producers and consumers connect to a server, which acts as router for disseminating and dispatching events.

Several routing protocols aimed at providing end-to-end QoS in ad hoc networks have succeeded in providing soft real-time guarantees or in reducing transmission latency but with no further guarantees. Proactive and preemptive routing [20] attempt to seamlessly switch to a good route before a link failure occurs minimizing both the transmission latency and the jitter, which is essential for real-time communication. The TBMAC protocol [21], reduces the probability of collisions by providing each wireless node with time-bounded access to the medium with a high probability. In addition, TBMAC uses an admission test to limit the impact of changes in network density on the volume of collisions. Therefore, it is suitable for providing predictable medium access latency for real-time event-based communication. Other solutions assume that time bounds may vary to maintain a guaranteed probability of success restricting the approach to those applications that can adapt time bounds, i.e., the time-elastic [22] class of applications which do not require hard real-time guarantees. Note that real-time systems dedicated to stationary environments and/or which rely on static scheduling are not in the scope of this paper. In the following section, we describe our solution dealing with the previously discussed challenges and requirements.

IV. Proposed Solution

Our approach provides a cross-layer architecture that supports the development of real-time context-aware applications. The overall architecture is illustrated in Figure 1, with a detailed description of each component being given in the following sections. On the application level, this architecture provides the sentient object model described in Section IV-A. On the middleware level, event-based communication between sentient objects is achieved via RT-STEAM with a detailed description being given in Section IV-B. Finally, on the routing layer, the space-elastic adaptation provides timeliness guarantees within specified proximities as described in Section IV-C.

A sentient object gathers context information from its environment and reacts by producing RT-STEAM. Our proposal builds on the assumption that the closer event consumers are located to a producer the more likely they are to be interested in the events that it produces. Significantly, this implies that *events are relevant within a certain geographical area surrounding a producer*. In the context of our proposal, we consider only events that are raised on a periodic basis

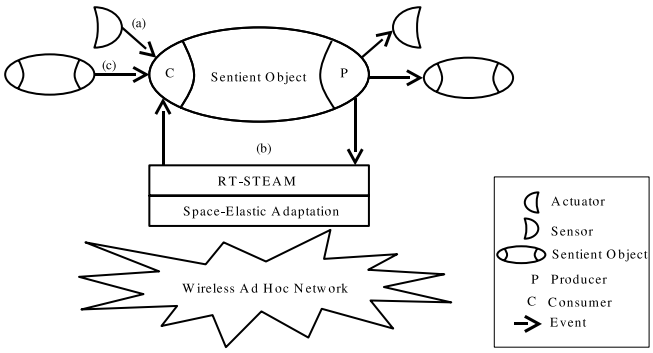

Fig. 1. System architecture

by their producer. Actually, a sentient object aimed at producing periodic real-time events will first define (*announce*) an implicit event channel as described in Section IV-B. An event channel specifies a combination of proximity and QoS attributes for delivering of events of some type. The proximity attributes allow the producer to bound the geographical area within which its events are relevant and within which the QoS attributes defining the real-time constraints of these events have to be maintained.

Sentient objects acting as consumers must subscribe to event types in order to receive subsequent events once located inside any proximity where events of this type are raised. A consumer may move from one proximity to another without re-issuing a subscription when entering the new proximity. A single subscription may result in events of a particular event type raised by different producers in multiple proximities and with different QoS being delivered over time. A consumer is said to be *present* for an event type once it has subscribed to this event type and is located inside a proximity where events of this type are raised.

A sentient object that produces a real-time event is guaranteed timely delivery of the event to all consumers which are *present* within the actually covered proximity (i.e., the proximity where QoS constraints for the event type are satisfied) at the event deadline. On the other hand, a consumer is guaranteed timely event delivery if it is *present* at the event deadline within the actually covered proximity.

A. The Sentient Object Model

Context-aware applications require the ability to perceive, interpret and react according to the state of the environment. In our model context-awareness is achieved through the notion of sentient objects. A sentient object is an encapsulated entity, with its interfaces being sensors and actuators. *Sensors* in the sentient object model are defined as entities that produce software events in reaction to a real world stimulus, and are

abstractions of physical transducers or software components. *Actuators* in the sentient object model are entities that consume software events and react by attempting to change the state of the real world in some way. A sentient object is then defined as an entity that can both consume and produce software events, and lies in the control path between at least one sensor and one actuator [23].

Each sentient object receives context information from potentially three sources, as following: (i) *sensors* provide information about the physical environment of the sentient object (Figure 1(a)); (ii) *infrastructure components* local to the sentient object provide information about the state of the computational infrastructure (Figure 1(b)); (iii) other *sentient objects* may also provide information to the sentient object (Figure 1(c)). The design of the sentient object model is component based, and therefore it is scalable and easily extensible. Sentient objects are composed of three major internal components jointly providing these functionalities:

1) **Sensory capture and context fusion.** Sensor data captured from the real world, as well as infrastructural information, are inherently noisy and unreliable. The sensory capture component fuses multi-modal data to determine higher-level context information. This component employs a probabilistic sensor-fusion scheme, based on Bayesian networks [24], to provide a powerful mechanism for measuring the effectiveness of derivations of context from noisy data.
2) **Context hierarchy.** The set of contexts in which a sentient object can exist is represented as a hierarchy, based on the Context Based Reasoning (CxBR) paradigm [8]. Each context in the hierarchy encapsulates knowledge about actions to be taken in that particular context, as well as what type of events are relevant in that context. Additionally, each context defines the other contexts to which a transition may occur.
3) **Inference engine.** The inference engine component uses conditional rules coupled with a knowledge base to control object behaviour. The current active context in the context hierarchy defines a subset of conditional rules that are valid at that point in time, which arises from the hypothesis that there are only a limited number of things that can realistically take place in any situation, and by limiting the number of actions permitted in specific contexts, the efficiency of production rule-based inference is substantially increased.

B. RT-STEAM

RT-STEAM implements an implicit event model [25] that allows event producers to publish events of severral event types and consumers to subscribe to one or more event types. RT-STEAM has been designed to support applications in which application components can be either stationary or mobile and interact based on their geographical location. This implies that the RT-STEAM middleware as well as the entities hosted by a particular machine are aware of their geographical location at any given time. RT-STEAM includes a location service that uses sensor data to compute the current geographical location of its host machine and entities. To suit outdoor applications, for example, in the traffic management domain.

In addition to supporting stationary and mobile entities, RT-STEAM allows proximities to be either stationary or mobile. A stationary proximity is attached to a fixed point in space whereas a mobile proximity is mapped to a moving position represented by the location of a specific mobile producer. Hence, a mobile proximity moves with the location of the producer to which it has been attached. This implies that mobile consumers and producers may be moving within a mobile proximity.

C. Space-Elastic Model

The space-elastic model assumes that real-time applications are space-aware. It combines a proactive, mobility-aware routing and resource reservation protocol, at the network layer, with a predictable time-bounded medium-access control (TBMAC) protocol [21]. TBMAC protocol is based on time-division multiple access with dynamic but predictable slot allocation. It uses a lightweight atomic multicast protocol to achieve distributed agreement on slot allocation and employs location information to minimise contension for slots. The resource reservation protocol, attempts to discover and maintain real-time constrained routes within a proximity, e.g., as specified by the space-elastic model, utilizing the cellular structure of TBMAC coupled with admission control to establish real-time routes spanning potentially multiple TBMAC cells.

A further assumption of the space-elastic model is that the proximity is defined within maximum and minimum bounds in order to support reducing the size, or changing the shape, to guarantee desired real-time communication. Therefore, proximity attributes defined by RT-STEAM proximity filters specify in addition to an adaptation notification handler, the maximum proximity to be covered, called the *Desired Coverage* (DC) which is used to limit the scope of event propagation; and the minimum coverage that will allow consumers to react

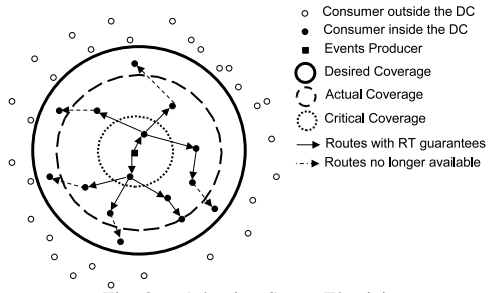

Fig. 2. Adaptive Space-Elasticity

safely to received events, called the *Critical Coverage* (CC). The proximity within which real-time guarantees are currently available is called the *Actual Coverage* (AC), as depicted in Figure 2.

Based on the network connectivity and the available resources, the space-elastic model adapts the *Actual Coverage* between the desired and the critical coverage in order to maintain real-time guarantees (See Figure 2). Ideally, the *Actual Coverage* should be equal to the *Desired Coverage*. On the other side, failure to guarantee the *Critical Coverage* is a real-time failure, the consequences of which and the actions to arise are determined by the associated real-time application and requires transition to fail-safe mode by the producer. Whenever the space-elastic adaptation adapts the *Actual Coverage*, the producer is notified in a timely manner. This adaptation feedback is considered as additional context information.

We have adopted a producer centric approach; therefore a consumer has no explicit feedback on any adaptation or changes happening in its surrounding environment. However, a consumer that delivers at least one event is expecting an event delivery of the same event type within a known time bound (event period). This knowledge allows the consumer to undertake appropriate actions if such a delivery does not happen.

The timely adaptation notification of the producer is reflected through the definition of the *Stability Period* (SP). The stability period is the minimum duration after event transmission when a producer knows that all consumers *present* at the event deadline have been delivered the event. Therefore, the guarantees provided to a producer regarding event delivery held after the duration of the *Stability Period* provided that no adaptation notification has been issued by the space elastic adaptation before the end of the *Stability Period*.

In the context of mobile real-time applications, the definition of the *Critical Coverage* is crucial to allow mobile consumers to react safely to delivered events and to allow the producer to switch to the fail-safe mode if the *Actual Coverage* falls below the *Critical Coverage*. Therefore, the *Critical Coverage* can be defined from two perspectives:

- From the consumer's perspective: as previously mentioned, the *Critical Coverage* must allow a mobile consumer to have enough time to react safely to the delivered event. Therefore, the *Critical Coverage* definition must take into account the delay for a consumer to receive events after arriving within the actual coverage (the time to be *Present*), the frequency of raising events (*Period*) and the consumer's reaction time (*C_Reaction*) which is application specific (for example, the braking time of an autonomous car).
- From the producer's perspective: following our model, hard real-time events are delivered at the specified deadline. A producer will know that an adaptation happened if it receives a notification within the *Stability Period* associated with each event (*AdaptNotif*). Therefore, the *Critical Coverage* must allow the producer to react to an adaptation by entering the fail-safe mode. To accommodate the worst case, the *Critical Coverage* definition includes the stability period and producer's reaction time (*P_Reaction*) which is application specific.

To reconcile both consumer's and producer's perspectives, the *Critical Coverage* can be defined as the distance travelled by a mobile consumer (at application specified speed) during the following duration:

$$Present + Period + max(C\_Reaction, \quad AdaptNotif + P\_Reaction) \quad (1)$$

V. Conclusions

Providing context-aware applications with hard real-time guarantees is challenging, and particularly if the underlying communication infrastructure uses wireless ad hoc networks. In this paper, we identified the main challenges and the requirements on different architectural levels in order to provide such guarantees.

Several research work has been carried out to provide adequate middleware for the development of context-aware applications. None of the existing systems has succeeded in providing adequate solutions to all the challenges and requirements. For instance, these systems provide only soft real-time guarantees, and only few of them actually support ad hoc networks. Therefore, we describe our solution which provides hard real-time guarantees in context-aware applications aimed at wireless ad hoc networks. Our proposal builds on three main components: sentient objects, an event-based real-time middleware, and a real-time routing and resource reservation protocol. It provides the following key functionalities: *(i)* it

relieves the application developer of the burden of low level interaction with various hardware devices; *(ii)* it provides timeliness guarantees in particular hard real-time within bounded geographic areas; *(iii)* it supports dynamic wireless ad hoc networks.

In order to perform real scale experiments and to show the advantages of using our architecture, we are working on the implementation of the aforementioned components. Some components of our architecture are implemented while the implementation of some others is part of our work in progress. Sentient objects are developed using a graphical development tool which allows developers to specify relevant sensors and actuators, define fusion networks, specify context hierarchies and production rules, without the need to write any code [23]. Based on the specifications provided by the developer, the programming tool generates the complete code for a sentient object including Bayesian network descriptions, context representations, event filters and CLIPS rules. The code generation currently provides output in C++, but the support for real-time applications is not yet included. RT-STEAM is an extension of the STEAM [25] middleware to support hard real-time delivery of events in mobile ad hoc networks. The STEAM middleware is implemented in C++ and provides all the functionalities required from an event-based communication middleware. RT-STEAM APIs have been defined; however their implementation is still an ongoing work pending completeness of an implementation of the TBMAC protocol with a real-time routing and resource reservation protocol.

VI. Acknowledgments

The authors are grateful to Science Foundation Ireland for their support of the work described in this paper under Investigator award 02/IN1/I250 between 2003 and 2007.

References

[1] T. Kindberg and A. Fox, "System software for ubiquitous computing," *IEEE Pervasive Computing*, vol. 1, no. 1, pp. 70–81, 2002.

[2] J. Kaiser and M. Mock, "Implementing the real-time publisher/subscriber model on the controller area network (CAN)," in *Proceedings of the 2nd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, 1999.

[3] H. Hartenstein, B. Bochow, A. Ebner, M. Lott, M. Radimirsch, and D. Vollmer, ""position-aware ad hoc wireless networks for inter-vehicle communications: the fleetnet project"," in *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking and computing*, 2001.

[4] A. K. Dey, D. Salber, and G. D. Abowd, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications," *Human-Computer Interaction Journal*, vol. 16, pp. 97–166, 2001.

[5] A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. V. Laerhoven, and W. V. de Velde, "Advanced interaction in context," in *Proceedings of 1st International Symposium on Handheld and Ubiquitous Computing*, vol. 1707. Springer-Verlag, 1999, pp. 89–101.

[6] P. Castro and R. Muntz, "Managing Context Data for Smart Spaces," *IEEE Personal Communications*, vol. 7, pp. 44–46, 2000.

[7] A. Ranganathan, J. Al-Muhtadi, and R. H. Campbell, "Reasoning about Uncertain Contexts in Pervasive Computing Environments," *IEEE Pervasive Computing*, vol. 3, no. 2, 2004.

[8] A. J. Gonzalez and R. Ahlers, "Context-based representation of intelligent behaviour in training simulations," *Transactions of the Society for Computer Simulation International*, vol. 15, no. 4, 1999.

[9] A. Ranganathan, R. E. McGrath, R. H. Campbell, and M. D. Mickunas, "Ontologies in a Pervasive Computing Environment," in *Proceedings of Workshop on Ontologies and Distributed Systems*, 2003.

[10] L. Kagal, T. Finin, and A. Joshi, "Trust-Based Security in Pervasive Computing Environments," *IEEE Computer*, vol. 24, no. 12, pp. 154–157, 2001.

[11] P. Osbakk and N. Ryan, "A Privacy Enhancing Infrastructure for Context-Awareness," in *Proceedings of* 1^{st} *UK-UbiNet Workshop*, 2003.

[12] *CORBAservices: Common Object Services Specification - Event Service Specification*, Object Management Group, 1995.

[13] *CORBAservices: Common Object Services Specification - Notification Service Specification*, Object Management Group, 2000, version 1.0.

[14] T. Harrison, D. Levine, and D. Schmidt, "The design and performance of a real-time corba event service," in *Proceedings of the Conference on Object-Oriented Programming Systems, Languages and Applications*, 1997.

[15] J. Orvalho, L. Figueiredo, and F. Boavida, "Evaluating light-weight reliable multicast protocol extensions to the corba event service," in *Proceedings of the 3rd International Enterprise Distributed Object Computing Conference*, 1999.

[16] J. Bacon, K. Moody, J. Bates, R. Hayton, C. Ma, A. McNeil, O. Seidel, and M. Spiteri, "Generic support for distributed applications," *IEEE Computer*, vol. 33, pp. 68–76, 2000.

[17] G. Cugola, E. D. Nitto, and A. Fuggetta, "The jedi event-based infrastructure and its application to the development of the opss wfms," *IEEE Transactions on Software Engineering*, vol. 27, pp. 827–850, 2001.

[18] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Transactions on Computer Systems*, vol. 19, pp. 283 – 331, 2001.

[19] P. Sutton, R. Arkins, and B. Segall, "Supporting disconnectedness transparent information delivery for mobile and invisible computing," in *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid*, 2001.

[20] T. Goff, N. B. Abu-Ghazaleh, D. S. Phatak, and R. Kahvecioglu, "Preemptive routing in ad hoc networks," in *Proceedings 7th Annual International Conference on Mobile Computing and Networking*, 2001.

[21] R. Cunningham and V. Cahill, "Time bounded medium access control for ad hoc networks," in *Proceedings of Principles of Mobile Computing*, 2002.

[22] P. Veríssimo and A. Casimiro, "The timely computing base model and architectures," *Transaction on Computers - Special Issue on Asynchronous Real-Time Systems*, vol. 51, no. 8, 2002.

[23] G. Biegel and V. Cahill, "A framework for developing mobile, context-aware applications," in *Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communications*, 2004.

[24] J. Pearl, *Bayesian Networks Handbook of Brain Theory and Neural Networks*. MIT Press, 2001.

[25] R. Meier and V. Cahill, "Exploiting proximity in event-based middleware for collaborative mobile applications," in *Proceedings of the 4th IFIP DAIS*, 2003.