A Component-based Middleware Architecture for Sentient Computing

Aline Senart, Mélanie Bouroche, Gregory Biegel and Vinny Cahill
Distributed Systems Group,
Department of Computer Science,
Trinity College, Dublin
{Aline.Senart, Melanie.Bouroche, Greg.Biegel, Vinny.Cahill}@cs.tcd.ie

Abstract

The Aithne project is developing a component-based middleware architecture for sentient computing that provides developers with the possibility of constructing customised middleware to support different combinations of non-functional requirements. The architecture that we are designing is based on the concept of *sentient objects* - mobile intelligent software agents capable of sensing their environment and reacting based on their perceived context. In this paper, we identify the software components required to support sentient objects in ad hoc environments and discuss an instantiation of the architecture for a *sentient sofa*, capable of recognising who is sitting on it.

1 Introduction

This last decade has witnessed explosive growth in the number and diversity of mobile, communicating devices, thanks to recent technological developments in wireless data communications and device miniaturisation. Based on these developments as well as new sensor technologies, a new class of applications, distributed over large geographical areas, and composed of a large population of fined-grained, context-aware and highly mobile entities, is emerging.

A typical example of such sentient computing application is traffic management. In such a scenario, entities may represent traffic lights, cars, ambulances or signposts, disseminating information on their context (e.g., location and state) and reacting to information provided by other entities in order to obtain safer driving. In the Aithne project, we model those intelligent entities as sentient objects [4], able to sense and to discover their context, to act autonomously (without direct human control) and to cooperate with each other.

The development and deployment of sentient computing applications requires middleware that supports a suitable interaction model with appropriate runtime services for sentient objects. However, existing middleware for context-aware computing does not meet the requirements of sentient objects, as it lacks support for large scale systems, time constraints, and mobility. Furthermore, customising middleware to fit application requirements is often not possible.

Given the diversity of potential applications constructed from large collections of sensor devices and the mix of requirements, we plan to provide a middleware architecture that describes a family of possible middleware platforms, addressing different combinations of non-functional requirements, like mobility and real-time behaviour. The architecture we are working on is based on a common programming model for sentient objects and can be customised by choosing from a library of reusable components.

As preliminary work, we have instantiated our middleware architecture for a sensor-augmented sofa, that identifies the person sitting on it by their weight and determines their approximate position. One possible application of the sentient sofa could be, for example, to set permissions on TV programs according to the person using the sofa. To develop the middleware for the sofa, we have selected the components required from our library and assembled them according to their well-defined interfaces.

In this paper, we identify the set of compon-

ents required by middleware for context-aware computing in ad hoc environments and present the application of those components to a sentient computing scenario. The structure of the paper is as follows. In the next section, we discuss related work. Section 3 and 4 present the design principles and components we have identified for our middleware architecture. Following this, the customisation of our architecture for the sentient sofa is presented in Section 5. Finally, concluding remarks and future work are outlined in Section 6.

2 Related work

Mobile nodes equiped with wireless interfaces and using ad hoc protocols are able to form a temporary network without the need for any established infrastructure or centralised administration. With the considerable research attention given to those ad hoc networks in recent years, new challenges for sentient computing arise: scarce resources, mobility, and time criticality are the most significant issues to be addressed. While there have been numerous efforts at developing context-aware applications [5, 13], relatively little work has been done to define appropriate middleware to support sensor technologies and to provide run-time services on top of ad hoc networks.

Traditional distributed middleware systems (like CORBA[1] or RMI[2]) are inadequate for the development of sentient systems: their synchronous communication model does not accommodate wireless applications. Moreover, those architectures are heavyweight in terms of memory and computation requirements and are therefore not suitable for sentient objects that can be characterised by scarce resources.

To provide lightweight middleware, several projects have attempted to adapt and scale their platforms to target applications [7], but they lack support for sensing and discovering their environment and have no ability to respond to the resulting contextual information. Furthermore, they do not provide support for different non-functional requirements, such as mobility or time constraints.

Current real-time middleware implementations [2] are appropriate for real-time applications, but can not be used in ad hoc networks. On the other hand, some middleware solutions provide system services to allow device mobility [9,14] and context-awareness [12], but do not address real-time issues. Other interesting systems have been developed to address real-time issues in wireless environments supporting ad hoc communication [15], but they fail to offer customable middleware where real-time aspects are optional.

To summarise, existing middleware described in the literature provides fixed and limited support for sentient computing. Existing solutions either provide a heavyweight environment or focus on specific issues, like user mobility or real-time guarantees. None of them provides a suitable framework that meets all sentient computing requirements. Moreover, the basic components we have identified to support sentient objects are not provided by related work: few systems offer this large set of components, ranging from a partition anticipator to an inference engine.

3 Design principles

The architecture that we aim to provide should enable the construction of customised middleware supporting different non-functional properties and target platforms.

3.1 Instantiation of customised middleware

Since resources available to sentient objects may be limited, the architecture should allow the construction of lightweight middleware. System services required by the applications for which middleware is targeted should be the only ones embedded within the middleware. For that purpose, we have identified the components that may be required to support sentient objects in ad hoc networks and the set of their interfaces (i.e., the services that these components make available to other components). Components are placed at the disposal of middleware designers in a library. This approach allows components to be assembled by developers to form customised middleware that fit application requirements exactly. All components offering the same service provide interfaces that conform a common specification, so their clients can use the service easily and can change the actual component that provides it. By using such

[1] http://www.omg.org
[2] http://www.java.sun.com

off the shelf components and well-defined interfaces, reusability and interoperability is greatly improved.

3.2 Support for non-functional guarantees

The middleware architecture we propose provides the necessary support for running sentient computing applications with non-functional guarantees. The support addressing non-functional requirements is formed by optional components available in the library. For example, the middleware architecture includes components to monitor timing constraints for safety critical systems. Likewise, runtime services for mobility are provided in the library for middleware dedicated to mobile applications.

3.3 Support for different platforms

Middleware for sentient computing needs to provide a uniform way to express context-awareness without restricting the application software developers to use a specific operating or communication system. Consequently, we provide a middleware architecture which can be easily adapted to different hardware environments, like embedded platforms, WinCE or linux. Moreover, simulations of large-scale sentient computing applications and deployment of context-aware applications in the real world are both possible. To achieve this, a simulator and different versions of components for different platforms are provided in the library.

4 Overview of the middleware architecture

We have identified the components that are necessary for the construction of middleware for sentient computing. In this section, we describe each of these components and their current implementations[3]. Figure 1 shows their organisation in different layers: a *programming model*, a *communication layer*, *routing and resource reservation mechanisms* and a *network layer*. We will now describe each of these layers.

[3]Due to space limitations, their interfaces are not presented here.

4.1 Programming model

First, we propose a programming model, based on the sentient object model [1], which provides components for the development of context-aware applications. The programming model allows data capture and fusion from disparate sensors, representation of application context, and efficient reasoning about context. A graphical development tool is also provided, allowing developers to specify relevant components to use in a sentient object, without the need to write complex code. The components of this programming model are:

Sensor capture and fusion. A sentient object may receive input from diverse sensors; thus, sensor fusion needs to be performed in order to manage uncertainty of the data coming from individual sensors and to determine higher-level context. For sensor fusion, a probabilistic scheme is provided in the library, based on Bayesian networks, as well as components performing a simple data sum and average.

Context representation. The set of higher-level contexts in which a sentient object may exist can be represented as a hierarchy. We place at the disposal of middleware developers a component for context representation that follows the Context-Based Reasoning paradigm [6], defining a set of possible actions and identifying future contexts according to the current context.

Inference engine. Sentient objects are expected to act upon their environment by using reasoning mechanisms that specify their behaviour in regard to the set of facts generated in their current context. An inference engine component that offers the conditional rule mechanisms of CLIPS[4] is available in the library.

4.2 Communication layer

Middleware supporting event-based communication and membership management are widely recognised as being appropriate to ad hoc networks since they naturally accommodate a dynamically changing population of sentient objects [10].

[4]http://www.ghg.net/clips/CLIPS.html

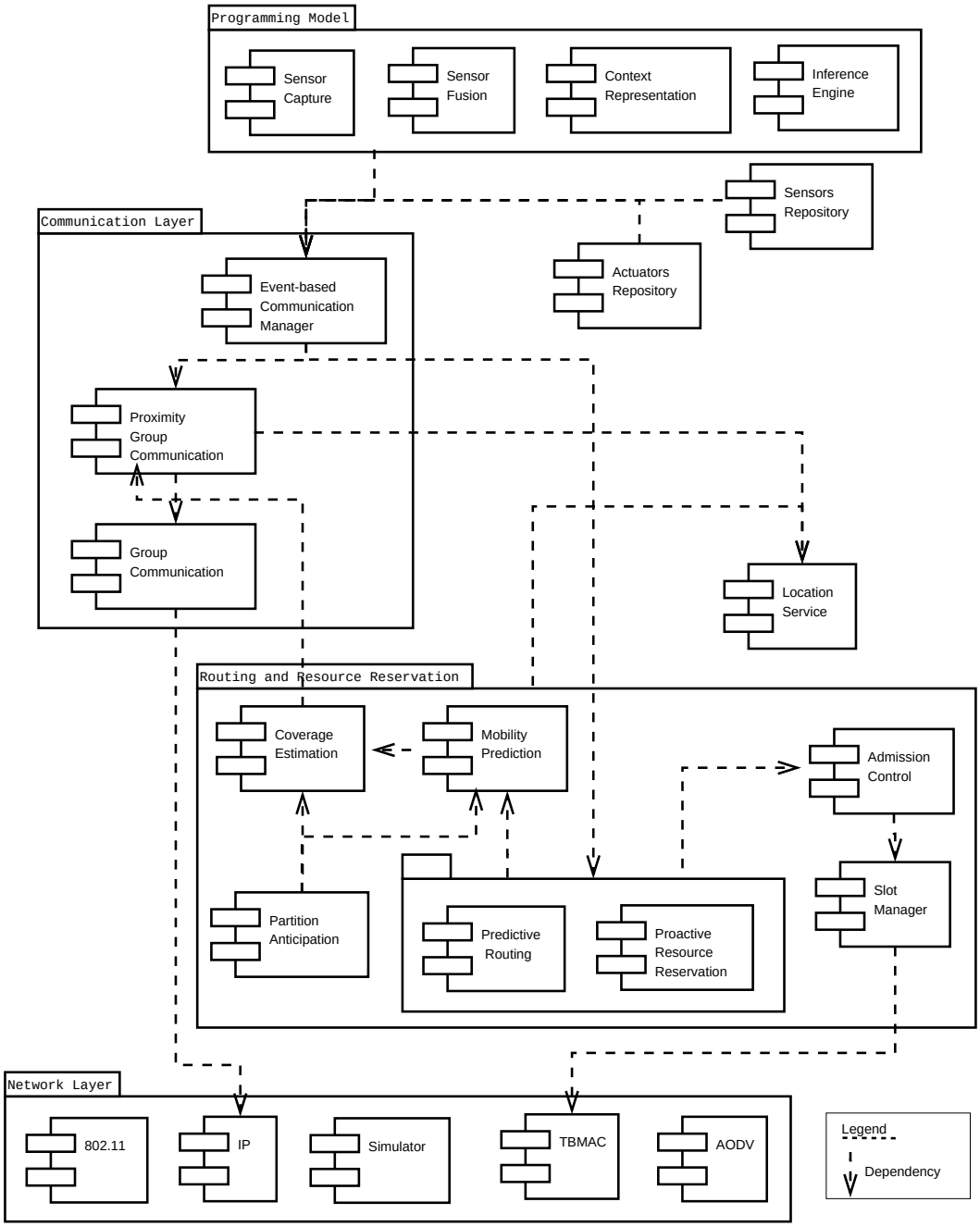

Figure 1: Components for sentient computing

Event-based communication. An event-based communication manager that does not depend on any fixed communication infrastructure is required for mobile applications in ad hoc networks. STEAM [11] is an implementation of a distributed event-based service that allows unpredictable and dynamic interactions between sentient objects. It provides filters that can be applied to the subject and content of event notifications, or to the geographical area within which event notifications are valid.

Membership management. As sentient objects share consistent information amongst each other (i.e., their context), a consistent view of objects involved in the communication is required. Classical group communication provides reliable multicast protocols that allow a producer to propagate events to a group of subscribed consumers. We have augmented group communication systems with the notion of proximity to capture the geographic location of mobile sentient objects. Only the objects present within the defined proximity can participate in the group communication. At the moment, we have a very simple toolkit for proximity-based group communication and classical group communication.

The sentient objects that form a group would cover a certain geographical area, that may be determined by a coverage estimation component. Information about the coverage area and mobility patterns may been employed to build a component which anticipates partitions and provides such information to applications. At this point, only partition anticipation is supported.

4.3 Routing and resource reservation mechanisms

To achieve real-time event-based communication in a mobile and ad hoc network, the unpredictability of the environment must be reduced [8]. Location-awareness is a key to predicting the future location of mobile sentient objects, easing proactive routing and resource reservation and then reducing the uncertainty inherent in such a dynamic environment.

In order to offer real-time guarantees, our architecture proposes a set of components realising proactive routing and resource reservation above the network layer. A slot manager component can allocate slots of the medium access in advance. A mobility prediction component, based on the location service and on the coverage estimation components may improve the routing process by attempting to find new paths prior to the failure of existing ones. Finally, an admission control component is needed in order to define policies that can be applied explicitly within a proximity to further reduce the number of participating sentient objects. We are currently developing those components.

4.4 Network layer

A number of network infrastructures available in the library can be used to support communication among sentient objects, like the WiFi protocol for wireless environments. We propose also TBMAC [3], a new MAC layer protocol that reduces the probability of collisions by providing each wireless node with time-bounded access to the medium with a high probability. We plan to use TBMAC to provide predictable medium access latency for real-time communication.

4.5 Additional components

Sensors and actuators. Several sensors have been developed as software components that produce software events, while actuators have been developed as software components that consume software events. These library components act as wrappers for hardware and software components as they provide mappings between specific data formats and generic events.

Location service. Location information is useful for computing devices in ad hoc networks. With the location service, we allow data from a variety of sources to be combined to generate information on location. Convertors are also available to modify location data formats.

5 Customised middleware for a sentient sofa

The sentient sofa[5] is a psychiatric sofa that identifies the person sitting on it by their weight. The sofa announces via a voice generator in which quarter of the sofa this person is located,

[5] http://www.dsg.cs.tcd.ie/?category_id=350

and the name of the person if it has been registered, and otherwise asks the person for their name. To evaluate the applicability of our middleware architecture to a context-aware scenario, we have instantiated it for the sentient sofa, using a subset of the components provided in our library. The sentient sofa is a simple application that encompasses some key characteristics of sentient computing, including the ability to perceive the state of the environment via sensors. The middleware dedicated to this application is composed of the following components:

Sensors and actuators. The sofa is augmented with load sensors placed underneath each of its legs and connected to a CPU by a serial connection. A load cell is a software sensor component defined in the library, able to continuously read information from the serial port and to generate events according to the stability of the readings. By tracking the readings variations and averaging several readings, the load cell can remove some errors. A voice generator is also used as an actuator that announces events happening on the sofa (e.g., a registration request or a movement on the couch).

Programming model. Sensor fusion consists of summing the four load sensors readings in order to determine the total weight of the person (each reading corresponding to the weight applied on one of the legs of the sofa). For this simple task, we have used the data sum component. The location is then determined by comparing each sensor reading with the average of the four readings: a person is said to be in one quarter if the load measured by the corresponding sensor is greater or equal to the average of the loads read on all sensors. A context hierarchy for the location of the person is defined according to the quarters of the sofa, with the possibility that the person can be in several quarters (e.g., when lying on the sofa). Finally, the inference engine contains rules to trigger events according to the current context. We use the CLIPS inference engine component to specify the facts and the rules of the sentient sofa.

Communication layer. To support the programming model, we use the STEAM component that allows an event-based communication between sensors, the sentient sofa and the voice generator. The different sensors in the application do not interact, so membership management components are not included in the middleware. Furthermore, routing and resource reservation mechanisms are omitted since timeliness of communication is not critical in this scenario.

Network layer. As the set of components is fixed in this application without mobility requirements, none of the components provided for ad hoc networks is used.

The construction of this middleware dedicated to the sentient sofa was easy thanks to off the shelf components of our middleware architecture. More specifically, using our architecture allowed a reduced development time.

6 Conclusion and future work

In this paper, we have described our preliminary work in defining component-based middleware for sentient computing. Instead of one single middleware, we propose a complete architecture that can be customised to various requirements. An initial list of components required by sentient objects has been presented. Some of those components have been assembled to form middleware for a sentient sofa. This experiment shows the feasibility of our proposal and the improvement in development time to build middleware for sentient computing. Future work includes the development of a formalism for a global description of middleware. The formalism could be associated with different tools, e.g. tools for graphical visualisation and composition, simulation, code generation, deployment or reconfiguration. We plan also to develop the library and in particular to integrate legacy communication models. Finally, we are interested in defining composite components for a better reusability at a coarse-grained level.

Acknowledgments

The work described in this paper has been realised as a part of Aithne project, which is founded by the Science Foundation Ireland (SFI).

References

[1] G. Biegel and V. Cahill. A framework for developing mobile, context-aware applications. In *2nd IEEE Conference on Pervasive Computing and Communications, Percom 2004*, pages 361–365, Orlando, Florida, March 2004.

[2] U. Brinkschulte, C. Krakowski, J. Riemschneider, J. Kreuzinger, M. Pfeffer, and T. Ungerer. A microkernel architecture for a highly scaleable real-time middleware. In *Real-Time Technology and Applications Symposium (RTAS 2000) - Work in Progress Session*, Washington, DC, May/June 2000.

[3] R. Cunningham and V. Cahill. Time bounded medium access control for ad hoc networks. In *Workshop on Principles of Mobile Computing (POMC'2002)*, Toulouse, France, October 2002.

[4] A. Fitzpatrick, G. Biegel, S. Clarke, and V. Cahill. Towards a sentient object model. In *Workshop on Engineering Context-Aware Object Oriented Systems and Environments (ECOOSE'2002)*, Seattle, Washington, November 2002.

[5] H. W. Gellersen, A. Schmidt, and M. Beigl. Multi-sensor context-awareness in mobile devices and smart artifacts. *Mobile Networks and Applications*, 7(6):341–351, October 2002.

[6] A. J. Gonzalez and R. Ahlers. Context-based representation of intelligent behavior in training simulations. *Transactions of the Society for Computer Simulation International*, 15(4):153–166, March 1999.

[7] J. Helander and A. Forin. MMLite: a highly componentized system architecture. In *Proceedings of the 8th ACM SIGOPS European workshop on Support for composing distributed applications*, pages 96–103, Sintra, Portugal, September 1998.

[8] B. Hughes and V. Cahill. Achieving real-time guarantees in mobile wireless ad hoc networks. In *Real-Time Systems Symposium (RTSS'03) - Work In Progress Session*, pages 37–40, Cancun, Mexico, December 2003.

[9] P. Kang, C. Borcea, G. Xu, A. Saxena, U. Kremer, and L. Iftode. Smart messages: A distributed computing platform for networks of embedded systems. *Special Issue on Mobile and Pervasive Computing, the Computer Journal*, to appear.

[10] R. Meier. Communication paradigms for mobile computing. *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R)*, 6(4):56–58, October 2002.

[11] R. Meier and V. Cahill. Exploiting proximity in event-based middleware for collaborative mobile applications. In *4th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'03)*, pages 285–296, Paris, France, November 2003.

[12] M. Roman and R. H. Campbell. A middleware-based application framework for active space applications. In *ACM/IFIP/USENIX International Middleware Conference*, pages 433–454, Rio de Janeiro, Brazil, June 2003.

[13] C. Shen, K. Everitt, and K. Ryall. Ubitable: Impromptu face-to-face collaboration on horizontal interactive surfaces. In *Fifth International Conference on Ubiquitous Computing (UbiComp 2003)*, pages 281–288, Seattle, Washington, October 2003.

[14] J. P. Sousa and D. Garlan. Aura: An architectural framework for user mobility in ubiquitous computing environments. In *3rd Working IEEE/IFIP Conference on Software Architecture*, pages 25–31, Montreal, Canada, August 2002.

[15] S. S. Yau and F. Karim. Context-sensitive middleware for real-time software in ubiquitous computing environments. In *Fourth International Symposium on Object-Oriented Real-Time Distributed Computing*, pages 163–170, Magdeburg, Germany, May 2001.