A Context-Aware Middleware for Applications in Mobile Ad Hoc Environments

Carl-Fredrik Sørensen[*], Maomao Wu, Thirunavukkarasu Sivaharan, Gordon S. Blair, Paul Okanda, Adrian Friday, Hector Duran-Limon

Computing Department, Lancaster University
Bailrigg, Lancaster, LA1 4YR, U.K.
Tel: +44 (0) 1524 593315
email:carlfrs@idi.ntnu.no, {maomao, t.sivaharan, gordon, okanda, adrian, duranlim}@comp.lancs.ac.uk

ABSTRACT

Novel ubiquitous computing applications such as intelligent vehicles, smart buildings, and traffic management require special properties that traditional computing applications do not support, such as context-awareness, massive decentralisation, autonomous behaviour, adaptivity, proactivity, and innate collaboration. This paper presents a new computational model and middleware that reflect support for the required the properties. The sentient object model is proposed by the CORTEX [1] project to support the construction of ubiquitous applications. A flexible, run-time reconfigurable component-based middleware has been built to provide run-time support to engineer the sentient object programming paradigm. An application infrastructure using sentient objects to enable cooperation between autonomous and proactive vehicles has been implemented to demonstrate the appropriateness of the computational model and the validity of the middleware for pervasive mobile ad hoc computing.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques—*Modules and interfaces*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

Keywords

Middleware, components, context-awareness, sentient objects, ad hoc wireless network

[*]Department of Computer and Information Science, Norwegian University of Science and Technology (NTNU). NO-7491 Trondheim, Norway

[1]This paper is part of the CORTEX (CO-operating Real-time senTient objects:architecture and EXperimental evaluation) project. The CORTEX project is supported by the EC, through project IST-FET-2000-26031. http://cortex.di.fc.ul.pt

2nd Workshop on Middleware for Pervasive and Ad-Hoc Computing Toronto, Canada

1. INTRODUCTION

The technical evolution in wireless networks, mobile and sensor technology has made the vision of ubiquitous computing coming closer to reality. Autonomous applications that in a proactive manner operate independently of direct human control are increasingly being developed to provide a more seamless and invisible support to the users. One of the challenges of this evolution has been to create ”intelligent” middleware and to design appropriate computational models for this new generation of applications. Such middleware should address the basic challenges raised by wireless mobile ad hoc network (MANET) applications, and support the growth and adaptability to new technologies, and provide applications with ways to enforce non-functional quality attributes like reliability and timeliness.

Academia and industry are both showing interest in research to exploit the new possibilities opened by the technological evolution. Car manufacturers, e.g., have initated projects to develop similar applications as investigated in this paper, e.g., the ConnectedDriv project by BMW [8] and the Intelligent Transportation Systems (ITS) program at the General Motors [1]. Such complex real-time dependable applications are difficult to engineer using traditional systems development techniques and paradigms.

The CORTEX project has identified the key characteristics that the new generation of applications may possess: **Sentience** – the ability to perceive the state of the surrounding environment; **Autonomy** – components should be capable of acting in a decentralized and autonomous fashion; **Time criticality** – applications will typically interact with the physical environment, and will have to cope with timeliness constraints; **Safety criticality** – typical applications will interact with human users, whose well-being will frequently rely on them; **Geographical dispersion** - typical applications will integrate components that are scattered over buildings, cities, countries, and continents; **Mobility** – applications residing in mobile devices must possess the ability to physically move and discover new neighbours and interact and share information; **Evolution** – applications will have to cope with changing conditions during their lifetimes.

It is unlikely that any application at the same time will possess all these characteristics. The most challenging applications are probably those that mix autonomy of application participants, derived from sentience, with the need to maintain a consistent view of the application environment while cooperating with other participants.

In this paper, we present a middleware platform which addresses the challenges raised by the application domain involving autonomous mobile physical objects that cooperate with other objects, either mobile or static, by capturing information in real-time from sen-

sors event messages propagated in a MANET.

The rest of the paper is organised as follows: Section 2 presents the application domain, Section 3 presents the sentient object model, and Section 4 presents the component framework and the middleware developed. Section 5 presents related work, and finally, Section 6 concludes the paper.

2. COOPERATING CARS

To investigate the unique challenges posed by ubiquitous applications built upon MANETs, we decided to build an application based on intelligent vehicles. This facilitates to investigate the feasibility and appropriateness of the programming model and middleware platform based on a concrete prototype application. Using real hardware for the demonstration ensures that our proposals are directly usable and do not depend on hidden idealistic assumptions. Moreover, the intelligent vehicle applications are an active research area. Cars are increasingly being equipped with different types of embedded sensors ranging from position and obstacle sensors to sensors indicating road and weather conditions. Cars with embedded sensors thus provide a rich set of context information. By making the sensor information available to other cars, a car can derive certain traffic conditions directly from the provided sensor information. Car-to-car communication thus has three major goals: *1. Dissemination of traffic information derived from the embedded sensors*; *2. Cooperation of cars to assist the driver in critical situations*; and *3. Interaction between remote cars*.
Traffic scenarios are inherently safety critical. An ultimate level of predictability and safety is particularly necessary in the tight coordination tasks in and between the cars. To model various forms of the aforementioned cases and to validate the middleware platform using real hardware, we have constructed a small number of autonomous robot cars. The robot cars have been instrumented with GPS[2], compass, and ultra-sonic sensors[3] and it is controlled by an iPAQ mounted on the car. Sensor information is collected by the iPAQ and used to provide actuations, e.g., change of course, reduction of speed, and in the case of obstructions or other safety critical events, to stop the car. Relevant information is notified to other cars using the ad hoc mode of IEEE 802.11b WLAN network.

3. SENTIENT OBJECT MODEL

The CORTEX project proposes the sentient object (SO) programming model for pervasive and ad hoc computing applications. The SO programming abstraction allows developers to design distributed applications in terms of sentient objects, instead of decomposing them into components parts such as messaging protocols, sensor fusions, context representation and inference engine. Those applications that have already been built has shown to facilitate good application design and hide the complexity from the application designer. The cooperating car demo application is designed using the sentient object model as described in [6]. A sentient object (SO) is a mobile (not mobile code), intelligent software component that is able to sense its environment by consuming events via event channels from sensors and/or other SOs, fuse the sensor data and derive higher level context, perform context-based reasoning using some control logic and actuate physical aspects by publishing events via event channels to respective actuators. SOs are context-aware; aware of both their internal state and the state of their local environment; and are cooperative by exchanging higher level context information via event channels.

[2]A Trimble Lassen LP GPS module is used in our cars.

[3]Thanks to Joe Finney and Angie Chandler for construction of the cars augmented with a token ring network and an array of sensors.

4. COMPONENT FRAMEWORK OF SENTIENT OBJECTS

Sentient objects (SO) provide an abstraction for very common behaviour in pervasive ad hoc computing: Sensing and viewing the behaviour of neighbour entities and based on a RT-Image of the environmental context, reason about it, and actuate environmental objects. E.g., a car modelled as a SO can view the location of its nearest car or a traffic light signal state, reason about it, and then actuate or alter its navigation. This behaviour involves spontaneously discovering neighbour nodes and interacting with them to share context data, and possibly to cooperate with each other to attain common goals (e.g., cars cooperate to avoid collisions). The SOs capture useful, light weight mechanisms that match the essence of many distributed computing mechanisms in pervasive ad hoc computing domain. For the application designer to address a specific domain in terms of SOs, an underlying goal of the middleware is to provide a suitable interface that provides the basic functionality identified in Section 3. We have designed and implemented a component framework-based middleware platform to provide the run-time support for the SOs to fulfil this goal. The middleware consists of several component frameworks (CF) where each of the CFs addresses certain research areas. The *Publish-Subscribe CF* is used for discovery of mobile entities in its proximity, communication and data sharing data among distributed entities forming MANETs. The *Group communication CF* is used to provide a group communication protocol suite that provides the support to route events in mobile ad hoc networks. The *Context CF* provides the facility for sensor fusion and the inference engine. The *QoS management CF* arbitrates the allocation of resources and provides facilities for monitoring and adaptating the QoS. The *Timely Computing base (TCB)* [3] is used to monitor the QoS of the event channels operating over MANETs; The goal of the middleware platform is to provide support for CORTEX applications and to support the sentient object (SO) abstraction. Figure 1 shows the SO model of the cooperating car demo application. In this model, in each car, the external environment is sensed by different sensors and consumed by the SOs as sensor readings. The execution platform (iPaq) and the wireless network are in the model also regarded as external environments by the SOs. The external environment produces environmental events that can be captured by the sensors. The sensor readings are sent as internal events to the subscribing SOs. The SOs can also subscribe to events from other SOs. Such events are received as external events. Two types of SOs have been modelled in our cooperating car application: The *car SO* and the *QoS Management SO*. Both receive events from "sensors" or virtual sensors, and actuate by sending internal events to the other SOs which would trigger adaptation of a certain component configuration, and to the actual actuators (e.g. the speed control).

Previous experiences in construction of reflective middleware [2] have motivated us to use reflection, component technology and component frameworks (CF) to create a middleware platform for MANETs. In pervasive mobile ad hoc networks, a dynamic environment is the norm rather than the exception. This mandates the middleware platform to be deployment time configurable and runtime reconfigurable. Clearly the demands of a given application and its associated environment will vary over time and situation, both in terms of fine grained adaptation (tuning parameters), and at a courser grained level (e.g. switching components). To support this level of configuration and reconfiguration, we have chosen to build our middleware based on the reflective component model OpenCOM [5]. This technology allows introspection of the running system and adaptation of any aspect of the system at run-time,

permitting ultimate flexibility. OpenCOM is based on Microsoft COM, and is adaptive and reconfigurable. The middleware platform is implemented for Windows CE

4.1 Context CF

The *context CF* in Figure 1 provides the functionality of the control engine of a sentient object and consists of a sensor fusion and an inference engine component. It offers gaussian modelling algorithms and dead-reckoning functionality. The *context CF* subscribes to events from the sensors and possibly from other sentient objects, and fuse the multi-source sensor data and derives higher level context, the inference engine constitutes the expert logic which reason about context and take autonomous decisions. The decisions are in-turn published to respective actuators and sentient objects.

4.1.1 Sensory Capture and Fusion

Uncertainty is a major problem while sensing the environment due to the inherent limitations of sensors with respect to accuracy and precision. Several methods have been explored to handle this problem. Statistical methods, such as Gaussian modeling and maximum likelihood estimation can give more accurate estimation of target values from the imprecise raw sensor data; dead-reckoning can be used if there is insufficient traffic or delays from the sensor capture; Bayesian networks are also proposed to do sensor fusion and to measure the effectiveness of other context derivation methods given inherently uncertain and noisy sensor data. The combination of low-level sensors to a sensor detecting more complex events is denoted as sensor fusion, the resulting sensor is called a virtual sensor. It should be noted that the correlation of multiple sensors has to be performed in the receivers, which requires the general availability of sensor information.

A widely-used method in sensor fusion and machine learning is the Gaussian modelling and multivariate Gaussian modelling. Raw data samples from the sensors at certain target values need to be gathered to establish some multivariate Gaussian distribution functions. Ultrasonic sensor readings were collected at certain target values, i.e. various distances from the sensors to the obstacle ahead. When new readings arrive while the car moves, they are fed into the established Gaussian functions, and the highest value of the current readings will be the most probable target value. We can thus obtain the most probable distance to an obstacle using Gaussian modelling. This technique was used in the car demo to detect pedestrians and non-event publishing obstacles within 3m range from the car. The ultrasonic sensors on the robot car had the capability to detect objects within 3m range.

Dead-reckoning techniques were used to compensate for insufficient data and latency in transmissions. Instead of relying on the latent current sensor readings for fusion, historical data is processed during run-time to generate more reliable and real-time values. This technique was especially useful in the car application for sensors, such as GPS which have unpredictable real positional update delays. The cars were able calculate their own 'dead-reckoned' position and update their own trajectory using extrapolation algorithms to compensate for the GPS latency.

4.1.2 Inference Engine

The autonomous behaviour of a SO is supported by an instance of an inference engine embodied in the SO. Low-level context, either directly derived from sensor readings or received through the event channels is fed into a rule-based inference engine to derive high-level context according to some rules specified in the rule-base inside the inference engine. When high-level context has been derived, the conditional rules (specified by the application programmer) are used to reason about the context and execute external functions that have been defined by the user to do e.g. actuations. Context data is represented as "facts" within the CORTEX inference engine. Internally, SOs can interact with the inference engine by asserting and retracting these facts. A rule-based inference engine CLIPS - C Language Integrated Production System [9] has been chosen as a base for the inference engine. CLIPS provides a cohesive tool for handling a wide variety of knowledge with support for three different programming paradigms: rule-based, object-oriented and procedural. Using the rule-based programming paradigm provided by CLIPS, rules can be specified to generate high-level context from fused sensor data or derived low-level context. CLIPS also provides ways for the programmers to define their own user functions, and these functions can also be called from the CLIPS rules script file. This makes it possible to automatically perform actuations when a certain context is derived from the inference engine. One of the important features of our approach is that the paradigm facilitates uniform treatment of both context and QoS data. E.g, CLIPS script based rule files are written by the application programmer to control the navigation of the cars in the Car SO. Similarly QoS management policies are written in CLIPS script file for the QoS management sentient object.

4.2 Publish-Subscribe CF

The key to the implementation of a sentient object is the publish-subscribe (P-S) communication model used for both discovery of neighbouring nodes and data sharing. When a SO wants to share its context data, the data is published using the P-S communication model. The receiving SOs subscribe to the events of interest. The P-S CF takes care of event routing and event filtering from a publisher to subscribers. This decouples the publisher and the subscribers, they are anonymous and the communication style is asynchronous. These properties are well suited for MANETs. However, the state-of-the-art P-S middleware are based on centralised event brokers that mediate between publishers and subscribers. The MANETs cannot support centralised fixed servers or fixed system wide services. Thus, we have specifically designed the P-S CF for MANETs. The design was inspired by STEAM [7]. The P-S CF is based on an implicit event model. The P-S CF differs from other P-S systems in that it does not rely on the presence of any separate infrastructure and supports distributed techniques for identifying and delivering events of interest based on location. The P-S CF support both publisher and subscriber side event filtering, by using a query or subscription language called Filter Event Language (FEL) to express the preferences. FEL can be used to create subject, content, and context filters, which are also componentised and can be dynamically reconfigured. In the implicit event model, there are thus no event brokers or mediators; instead event filtering functions are implemented at both the subscriber and the publisher side in a decentralised manner.

Publishers and subscribers are anonymous, and subscribers should be able to interpret the events without a priori knowledge of the exact event data structure. A generic event dialect is therefore needed, which can be understood by all publishers and subscribers in the system. The events in the P-S CF are represented in XML, and a generic XML profile defines the generic event dialect. The XML based events provide for easy interoperability and extensibility of the event dialect. Events are published to a notional event channel allowing one-to-many distribution of events. The events are routed from publisher(s) to subscriber(s) using multicast protocols. The multicast protocols are addressed in the *Group Communication (GC) CF*.

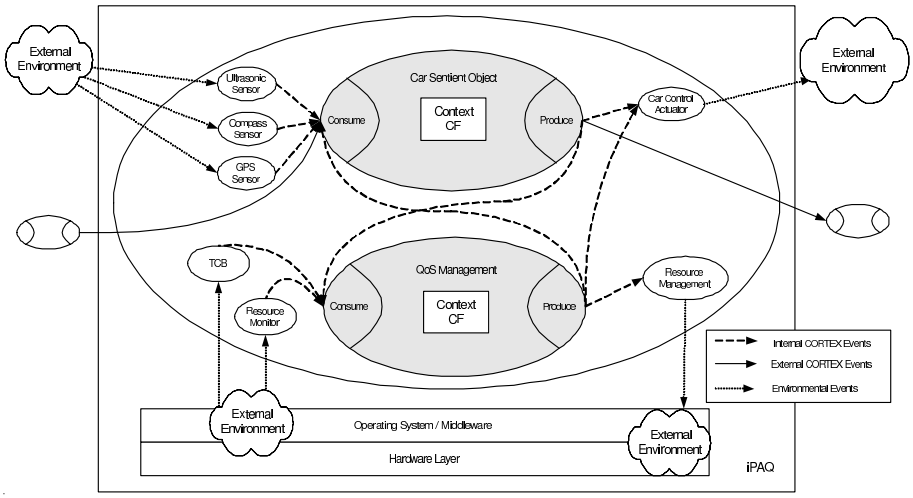

Figure 1: The Sentient Object Model of the Car Scenario

The *GC CF* provides a range of multicast protocols. The P-S CF can flexibly select a group communication protocol from the GC CF, which currently supports a probabilistic ad hoc multicast protocol, an IP multicast based protocol, and a local (shared memory) group communication protocol. The probabilistic ad hoc multicast protocol was implemented for MANETs with high node mobility, where keeping shared routing information within the nodes is unsuitable as the network topology is rapidly changing.

5. RELATED WORK

Different approaches of middleware in UbiComp have been investigated, e.g., Gaia and MobiPADS. Gaia [10] is a metaoperating system built as a distributed middleware infrastructure that coordinates software entities and heterogeneous networked devices contained in a physical space. Gaia supports development and execution of portable applications in *active spaces*. Users, services, data, and locations are represented in the active spaces, and are manipulated dynamically and in coordination. Context is important to Gaia's applications, which have access to context via a "context file system". However, context is externalised and is not a first class entity that drives the behaviour of the active space systems. We believe that active space applications could easily be modelled as sentient objects and potentially benefit from our paradigm.

The MobiPADS system [4] is a reflective-based middleware designed to support context-aware processing by providing an execution platform to enable active service deployment and reconfiguration of the service composition in response to environments of varying contexts. MobiPADS supports dynamic adaptation at both the middleware and application layers to provide flexible configuration of resources to optimize the operations of mobile applications. The reflective model in MobiPADS provides metainterfaces to make applications able to directly participate in computation adaptation in response to the changing context. A mobile application can access contextual information, the service configuration and adaptation strategy, and examine and modify these entities to obtain optimal service provision.

6. CONCLUSION

In this paper, we have presented a middleware for mobile ad hoc environments. The prototype implementation of the autonomus car application scenario provides a very rich application domain to explore context-awareness. The sentient object programming model has proven to be very usable as a programming abstraction, for development of such applications, particularly because of the intrinsic support for context-awareness. The context reasoning engine and middleware developed have successfully been used in an autonomous car application in a real mobile ad hoc wireless environment. There is a need for middleware in this area to ease the burden on the application developer and also to provide support for the management of non-functional concerns such as timeliness properties. The properties of configurability and re-configurability inherent in our approach are highly suited to this domain, for example to select configurations suitable for a given embedded device and also to encourage the construction of adaptable or autonomic systems. The middleware developed is reusable and may potentially also be (re)used in other application domains like smart buildings and retail systems. Investigating the generality and application of the approach in other domains is interesting for future work.

7. REFERENCES

[1] S. Ashley. Smart Cars and Automated Highways. *Mechanical Engineering Magzine*, May 1998.

[2] G. S. Blair, et.al. The Design and Implementation of Open ORB version 2. *IEEE Distributed Systems Online Journal*, 2(6), 2001.

[3] A. Casimiro and P. Veríssimo. Using the Timely Computing Base for Dependable QoS Adaptation. *20th IEEE Symposium on Reliable Distributed Systems*, pages 208–217, New Orleans, USA, Oct 2001.

[4] A. T. Chan and S.-N. Chuang. MobiPADS: A Reflective Middleware for Context-Aware Mobile Computing. *IEEE Trans. on Software Engineering*, 29(12):1072–1085, 2003.

[5] M. Clarke, G. S. Blair, et al. An Efficient Component Model for the Construction of Adaptive Middleware. *IFIP/ACM Middleware2001*, Heidelberg, Germany, Nov 2001.

[6] A. Fitzpatrick, et al. Towards a Sentient Object Model. *Workshop on Engineering Context-Aware Object Oriented Systems and Environments (ECOOSE)*, Seattle, WA, USA, Nov 2002.

[7] R. Meier and V. Cahill. STEAM: Event-based Middleware for Wireless Ad Hoc Networks. *Int'l Workshop on Distributed Event-Based Systems (ICDSC/DEBS02)*, pages 639–644, Vienna, Austria, 2002.

[8] Automotive Intelligence News. Talking Cars For Less Congestion – The Future Of Telematics. http://www.autointell-news.com/, 2003.

[9] G. Riley. CLIPS – A Tool for Building Expert Systems. http://www.ghg.net/clips/CLIPS.html, 2003.

[10] M. Román, et al. A Middleware Infrastructure for Active Spaces. *IEEE Pervasive Computing*, 1(4):74–82, Oct–Dec 2002.