CORTEX[1]: Towards Supporting Autonomous and Cooperating Sentient Entities

P. Veríssimo
U. Lisboa
pjv@di.fc.ul.pt

V. Cahill
Trinity College Dublin
vinny.cahill@cs.tcd.ie

A. Casimiro
U. Lisboa
casim@di.fc.ul.pt

K. Cheverst
U. Lancaster
kc@comp.lanc.ac.uk

A. Friday
U. Lancaster
adrian@comp.lanc.ac.uk

J. Kaiser
U. Ulm
kaiser@informatik.uni-ulm.de

ABSTRACT

A new class of application that operates independently of direct human control is starting to emerge. It is our belief that the development of such applications is highlighting the shortcomings of current communication architectures and middleware infrastructures. In particular, they do not adequately support advanced dynamic interaction models, e.g., in the field of autonomous agents, distributed AI, and mobile co-operating entities. As we describe, our work represents the beginning of an attempt to bridge the gap between the requirements being put on system support by these advances, and the shortcomings of current architectures and middleware models.

Therefore, this paper explores the problem of providing infrastructure support for large distributed systems composed of mobile autonomous components. It describes a programming model supporting these applications based on the concept of sentient objects, and a hierarchical distributed communication architecture. Because the components may be part of the physical environment, issues such as predictability and environment awareness in the interactions between objects and environment deserve particular attention.

1 INTRODUCTION

Human society, at every level, is increasingly dependent on information. Information systems such as the World-Wide Web are now massively pervasive and critical to the functioning of the global economy. We are now at the point where the emergence of a new class of large-scale *decentralised* and *proactive* applications, i.e., applications that operate independently of direct human control, can be envisaged. However, this is also where the demands put on system support by state-of-the-art programming models requiring highly dynamic interactions, continuous system evolution and predictable reaction to unanticipated situations, e.g., in the field of co-operating autonomous robots, distributed AI and advanced process control go far beyond the abilities of current architectures and middleware models. Most of the recent research on support for such applications has concentrated on the functional and behavioural characteristics of the participants (objects, agents, etc.), but surprisingly there has not yet been much attention given to the non-functional requirements put on the supporting substrate. Some characteristics we can anticipate include *autonomy*, *large scale*, *geographical dispersion*, *mobility* and *evolution*. We believe that in the near future highly distributed, autonomous, mission-critical computer systems will become ubiquitous and pervasive. It is likely that such systems will be built using networked components responding autonomously to a myriad of inputs, in order to affect and control the surrounding environment. This introduces additional non-functional characteristics of these applications, thereby giving rise to a difficult combination of requirements to be addressed: *sentience*, *time* and *safety*/*security* criticality.

CORTEX proposes to devise an architecture and a set of paradigms for the construction of applications composed of collections of what may be called *sentient objects* - mobile intelligent software components that accept input from a variety of different sensors allowing them to sense the environment in which they operate before deciding how to react. Furthermore, the latter may organise themselves into autonomous, mobile and rapidly composable co-operating communities. In the future, we expect that sentient objects will be pervasively included in almost every aspect of our daily life. They will ubiquitously integrate all kinds of devices and interact seamlessly amongst themselves in ways that go far beyond the client/server paradigm supported by current state-of-the-art middleware [11,13,21]. Applications will form islands of co-operation inside a wider network universe composed from different physical networks with characteristics ranging from high speed backbones to wireless connections and deeply embedded field buses.

In the long term, society will substantially rely on this technology. To reduce vulnerability and provide a robust failure resilient environment, middleware is required that understands the metaphors of the high-level models, yet keeps the underlying system (still a fragile computer and network system) within correct operational envelopes.

The remainder of this paper is structured as follows. In section 2 we analyse the key characteristics of the advanced applications mentioned above and the resulting problems. Sections 3, 4 and 5 describe the major layers of the *CORTEX* approach, the programming paradigm, the interaction model and the system architecture respectively, following a top-down system view. Section 6 sketches typical large-scale proactive applications that will require the advanced middleware support to be developed in *CORTEX*. Finally, section 7 presents our concluding remarks.

[1] This work was partially supported by the EU, under the IST/FET programme, through project IST-2000-26031 (CORTEX- CO-operating Real-time senTient objects: architecture and EXperimental evaluation).

2 FUNDAMENTAL CHALLENGES IN SUPPORTING SENTIENT APPLICATIONS

Fundamentally, *CORTEX* is concerned with helping to realise a vision of ubiquitous computing and proactive applications that are able to operate independently of direct human control. The approach of the project is to investigate the development of intelligent middleware capable of supporting appropriate computational models for this new generation of applications. This middleware must enable adaptability to new technologies, and provide the hooks for these applications to enforce non-functional quality attributes like reliability and timeliness. In particular, the middleware is intended to cope with applications that have *some or all* of the following characteristics:

- Sentience – the ability to perceive the state of the surrounding environment, through the fusion and interpretation of information from possibly diverse sensors;
- Autonomy – components of these applications will be capable of acting in a decentralised fashion, based solely on the acquisition of information from the environment and on their own knowledge;
- Large scale - typical applications may be composed of billions of interacting hardware and software components;
- Time criticality - these applications will typically interact with the physical environment, and will have to cope with its pace, regardless of adverse conditions due to scale and technology shortcomings;
- Safety criticality – typical applications will interact with human users, whose well-being will frequently rely on them;
- Geographical dispersion - unlike current embedded systems, typical applications will integrate components that are scattered over buildings, cities, countries, and continents;
- Mobility – furthermore, they must possess the ability to move between hosts possibly of different networks, while remaining in continuous operation
- Evolution – these applications will have to cope with changing conditions during their lifetimes. Not only must the applications be designed to evolve, but their underlying support must also be adaptable.

Traditional approaches to the design of time and safety critical distributed applications cannot handle the complexity inherent in the scale and geographic dispersion of these new applications.

However, whereas basic technologies exist that make autonomous decentralised systems a possibility, appropriate architectures and paradigms for the construction of the relevant applications are required. Consider applications composed of collections of *sentient objects*: they must be able to discover and interact with each other and with the physical world in ways that demand predictable and sometimes guaranteed quality of service (QoS), encompassing both timeliness and reliability guarantees, thus creating a fundamental trade-off between the existence of a dynamic environment and the need for predictable operation. To date, no comprehensive technology appropriate to the design and implementation of such applications exists.

Sentient objects will exist at very different levels of abstraction. At the lowest level, such objects might represent simple sensors or actuators capable of generating or consuming events. At a slightly higher level of abstraction, a tightly-coupled embedded system that integrates many such simple objects connected via a field bus might represent a single sentient object that is itself capable of generating and/or consuming events as a component of a larger system.

2.1 Communication, Co-ordination and Control

Irrespective of the level of abstraction at which we are working, three fundamental problems have to be addressed in order to support applications based on sentient objects: (1) dissemination of information to create common knowledge, mutual awareness and a basis for local decisions; (2) achieving co-ordination amongst peer objects in order to carry out actions in a consistent way; (3) acting upon the environment, changing its state as a result of proactive or reactive decisions.

Consider an advanced vehicular telematics scenario in which vehicles communicate with one another to provide a look-ahead warning service for vehicles coming from behind. If a vehicle detects an obstacle it sends an alert message that, in turn, the receiving vehicles can exploit in order to set new cruising parameters or brake as appropriate. In such a scenario, we face the following problems/challenges:

- The scope of information dissemination is dynamically determined by spatial parameters, i.e. those vehicles directly affected by the obstacle on the road.
- Communication is anonymous, hence group membership is implicit and reliable assessment of who received the message is difficult.
- The information is only valid in a restricted area.
- Many vehicles try to send similar messages, but the system should prevent the communication medium from being overloaded.
- Vehicles, which receive the message, must decide whether it is necessary to continue propagation or to stop.

The alert message would raise awareness between vehicles. The next step would be to initiate co-operation between the vehicles in order to enable vehicles to act in a coordinated fashion. While awareness may be realised as a best effort facility, co-operation needs a guaranteed quality of service between communicating entities. Given the above scenario, the underlying communications support must be able to deliver a wide range of qualities of service, in terms of both data exchange and membership services. State-of-the-art group communication protocols or generative anonymous communication [3] based on publisher/subscriber models definitely do not tackle these problems [20,23,24] and it is an open question whether these problems can be solved in the basic communication system alone.

2.2 Heterogeneity, Hierarchy and Scope

Again considering the example application outlined above, a hierarchy of communication networks will be present inside a vehicle to eventually convert the decision into deceleration or warning signals. Thus, the cruising parameters resulting from the higher-level co-ordination among vehicles have to be set and controlled by networks of intelligent sensors and actuators, that we generically call *controller area networks (CANs)*. In more general terminology, islands of control must co-operate via gateways in a timely and reliable manner, through the global wide area network (*WAN*). This mo-

tivates a crucial aspect of the *CORTEX* architecture, what we call a *WAN-of-CANs* structure.

The application example also unveils another important issue, which is related to the limit the range and control the quality of information propagation in the global system. Let us assume the notion of a *zone*. An important issue for co-operation is to establish what QoS can be sustained by the zones in which participants reside. Typically, a single CAN represents a zone with a very high level of predictability compared to a zone in a wireless network [15,19,25,27,32]. In a mobile environment where migration from one zone to another zone is likely to happen, it is a great challenge to devise communication mechanisms that dynamically adapt to these changing QoS attributes while maintaining a certain level of guarantee. Paradigms like zoning and topology awareness are relevant in our context, since they allow the heterogeneity of the underlying support to be accommodated, while not necessarily making it visible to the layers above [22,26].

2.3 Predictability and Adaptability

Underlying all of these considerations is the fundamental challenge of coping with the uncertainty of synchrony. In principle, this can be achieved by adaptation. However, while there is an increasing body of research on QoS adaptation [2], most work has focused on protocol or application-level heuristics and does not provide any guarantees on how well the system adapts. The applications we intend to support require *predictability* about timeliness. This means that even if the timeliness of the system is degrading, it should do so in a predictable way. In consequence, the coverage of timeliness assumptions should remain stable throughout the application's lifetime.

More demanding applications will require *guarantees* about timeliness objectives, that is, not only the coverage both also the assumed timeliness bounds should hold. Since timing faults are difficult to prevent in the kinds of complex and large-scale systems that we are considering, this presents us with a fundamental challenge of *avoiding contamination*, i.e. incorrect logical behaviour when timing faults do occur. This has been shown to be a significant problem even in systems where synchrony expectations are minimal. Ensuring timely system operation despite timing faults, on the other hand, requires timing fault tolerance mechanisms.

These are challenging problems in large-scale systems with uncertain synchrony, especially where wireless communication is employed. We intend to use and build on previous results on partial synchrony systems, such as the timed asynchronous and quasi-synchronous models [5, 28].

2.4 Scalability

Scalability represents a crucial transparency property concerning the ability to accommodate growth in a large-scale distributed system. Thus, connecting more participants to the system dynamically, including adding entire additional networks, or providing new services, should not be prevented by factors originating in the system design. The notion of *anonymous event-based computing* is central to addressing the needs of scalable systems in *CORTEX*. Nevertheless, supporting non-functional attributes like timeliness and reliability guarantees adds new and challenging dimensions to scalability.

CORTEX aims at providing appropriate abstractions to express awareness about the uncertainty and variations of physical message transmission. The recursive *WAN-of-CAN* concept and, on a higher level of abstraction, the notion of zones contribute to this goal. Furthermore, we allow applications to exploit this information proactively, e.g. by trading precision against timeliness of information. We address this problem in the context of a partial synchrony model, providing adaptation whilst ensuring stability of the coverage of timeliness assumptions.

2.5 Fault Tolerance and Security

'Real' systems derived from the *CORTEX* approach will require measures enhancing their dependability, both from the fault tolerance and the security aspects. The convergence of ubiquitous wireless and anonymous networking, and of powerful embedded computer systems, provides an interesting spectrum of computer devices and information repositories that poses challenging security and reliability problems.

While we give priority to problems concerning coordination, control, predictability, adaptability, and scalability, our architectural approach is in line with, and will easily accommodate the incorporation of, known and emerging paradigms in modular and distributed fault tolerance for both large-scale systems and small-scale real-time systems, and in cryptographic multiparty communication and processing.

3 PROGRAMMING PARADIGM

Mobile sentient objects have autonomous behaviour resulting from interactions with the physical environment, i.e. driven by sensor inputs, as well as from the internal state of the objects. Moreover, they must be able to discover and interact with each other in ways that may lead to unpredictable interaction patterns depending, for example, on their geographical proximity.

Fundamentally, the *CORTEX* programming model describes the facilities that will be provided to application developers responsible for the construction of *proactive applications* that employ mobile *sentient objects*. At the heart of the *CORTEX* programming model is an anonymous event-based communication model, which we discuss later. Using a non-blocking event-based model, we are able to achieve autonomous sentient behaviour that is independent of the problems associated with traditional blocking communication paradigms (such as RPC [1,10,14,18]). The programming model includes mechanisms for the specification of constraints on the propagation and delivery of events, and the means to express incremental real-time and reliability guarantees, in the form of QoS properties. QoS is taken as a metric of predictability in terms of timeliness and reliability.

From the programmer's perspective, the system model is therefore composed of *the environment* and *a set of sentient objects* that interact with it. *CORTEX* adopts the *active environment* metaphor. In more detail, the environment is part of the system, which, whilst being active, is responsible for disseminating information about its current state and/or events that take place in the actual physical environment to objects of the system. In addition, the environment may also be acted upon or modified by these same objects. Sentient objects are the active, mobile and autonomous entities in the system and are capable of taking decisions, and influencing both the environment and other objects. The programming model supports several different aspects of the behaviour of sentient objects including:

- acquiring information from the environment and other objects (the sentience aspect);
- *reacting* to possibly unexpected situations (the autonomy aspect);
- and *modifying* the state of the environment (the control aspect).

Unlike traditional distributed applications, sentient objects will often need to send messages to a set of other objects whose identities are not known to the sender and which can only be determined at the time that the message was actually sent. For example, an object may need to send a message to all the objects that are nearby at a given time.

While the basic concept of an event-based communication paradigm is simple and indeed hopefully intuitive, there are a number of difficult issues that need to be tackled if the paradigm is to be employed successfully in large scale proactive applications.

Filters provide a basic mechanism to allow objects to express interest, or lack thereof, in events of a certain type or containing certain combinations of parameter values. Essentially an object subscribing interest in events of a particular type should be able to provide a filter describing which occurrences of events of that type it wants to be notified of. Filters alone are, however, not sufficient. With filters an object may still receive notifications of occurrences of events in a part of the system with which it is not currently concerned.

Zones introduce a means of scoping or limiting the propagation of event notifications in the system. Objects can be organised into zones where a zone can be seen simply as a collection of objects and event notifications are only propagated within the zone of the object raising the event. Objects are organised into zones at the discretion of the application programmer based on functionality, geographical location or physical location on the network.

While filters and zones allow an object to specify, at some level, which event notifications it is interested in, they do not address non-functional requirements related to the delivery of such notifications. This will be achieved in the *CORTEX* programming model through the introduction of a generic means of expressing QoS properties encompassing timeliness and reliability, including consistency and ordering of event notifications.

4 INTERACTION MODEL

Interaction comprises the aspects of communication and coordination. An event-based programming model naturally leads to the spontaneous generation of messages rather than a request/response style of communication. This fact suggests the use of an anonymous generative communication model [3,23,24,7,17], using typed communication channels to connect producers and consumers of events according to a publisher/subscriber model. Autonomy is supported because the model does not force an explicit transfer of control, nor synchronization between producers of events and their consumers. Therefore, the interaction model of *CORTEX* is based on such a model reflecting the needs of object autonomy, system robustness and evolution. We extend and modify existing approaches in two major directions:

- we consider the fact that sentient objects not only communicate via the network but also, indirectly, through the environment, when they act on it. Thus the environment constitutes an interaction and communication channel and is in the control and awareness loop of the objects.
- we address new adaptable ways of guaranteeing temporal properties of interactions, in the presence of uncertain timeliness of the environment. It is intended to exploit context awareness of sentient objects to reach this goal.

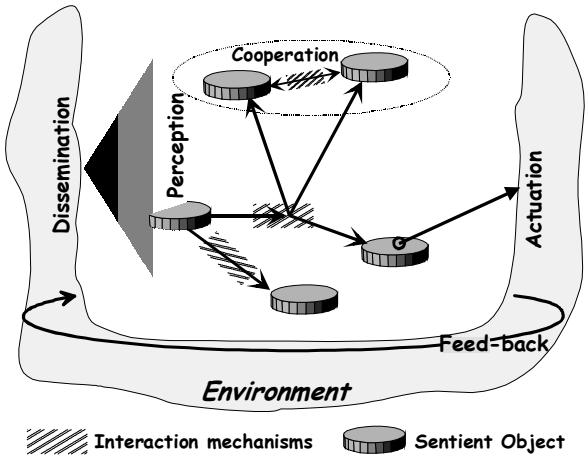

Figure 1: Events and object interactions in *CORTEX*

CORTEX must support several kinds of interactions (see Figure 1):

- **Environment-to-object interactions** take the form of unsolicited dissemination of the state of the former, and/or notification about events taking place therein. The transformation of events to state within the realm of the active environment components is not precluded, as a way to preserve the memory of past events.
- **Object-to-object interactions** serve two purposes. The first is related with complementing the assessment of each individual object about the state of the surrounding space, which includes environment components and the objects "within reach", that is, capable of influencing its next decisions. The second is related to collaboration, in which the object tries to influence other objects into contributing to a common goal, or into reacting to an unexpected situation.
- **Object-to-environment interactions** comprise the deliberate attempt at forcing a change in the state of the environment. This may come as a consequence of the pursuance of the object's own objectives, or of the reaction to unexpected situations created by the environment or other objects.

Given the highly interactive nature of the envisaged applications, and the fact that actions will be dictated to a great extent by assessment of the state of the environment, *CORTEX* falls under the typical constraints placed on distributed real-time systems [31,16]. Thus, the communication abstractions must support *predictable timing behaviour*.

CORTEX exploits *context and environmental awareness*, that is, use of internal and external context information to facilitate object interaction in changing situations. Objects, for example whilst moving, may be confronted with unpredictable communication needs or unanticipated inter-

action patterns with other objects and with the environment. Context awareness in terms of "which network am I in?", "how many hops away are my partners", "what delay am I to expect for this message", as well as the detection of timing failures or the assessment of membership all constitute examples of context awareness.

The main issue introduced by co-operation in the interaction model is the predictability of the *co-ordination mechanisms* necessary to carry out joint actions. Since objects operate in a real world environment, co-ordination has to be achieved under temporal constraints. As a minimum, this requires timeliness of communication, including those primitives achieving consensus, ordering, and so forth.

In order to address these issues, *CORTEX* combines the group communication and the anonymous communication paradigms in a flexible way, allowing non-functional properties such as the required degree of synchrony and the reliability of communication to be specified on a per group and per event basis.

Achieving predictable timing behaviour is a hard task in large-scale, heterogeneous systems that cannot be made strictly synchronous at reasonable costs in transmission delay and bandwidth. However, despite some of the adverse conditions just described, applications have to exhibit a certain degree of predictability. Approaches to this problem under uncertain operating conditions have been addressed in the mission-critical systems arena. Systems would normally have pre-defined operational envelopes, to which they would switch in a best effort to achieve their goal [12,29]. In more general terms, this is also the track followed by the QoS based adaptive systems. This adaptation is generally done in an ad-hoc manner, and may sometimes not bring the system to an optimal tuning.

In contrast, we address the hard problems in the time domain in the light of partial synchrony models, which can withstand varying timeliness or synchrony conditions, and the occurrence of timing failures. Our approach follows recent work on timing and QoS failure detection oracles [30] under partial synchrony models that reason in terms of the <assumption,coverage> binomial. This may help provide a precise definition of predictability, in terms of an assurance to which a probability is attached, and thus provide conditions for objects to make justifiable tradeoffs between maintaining their original goals with a reduced probability of success, or relaxing their goals whilst maintaining the initial probability.

5 SYSTEM ARCHITECTURE

The architecture of *CORTEX* must recognize two facts: much of the real infrastructure may actually be unknown prior to system deployment time, thus requiring the capacity for discovery of topology, services and so forth. Moreover, components are of an extremely heterogeneous nature, both in technological and exploitation terms (wired vs. wireless, public vs. private).

CORTEX features an *abstract network architecture* that reflects the hierarchical structure of large-scale heterogeneous networks, while defining the necessary mappings from this abstract description to real networks, including sensor/actuator busses and wireless links. In the architecture, non-functional properties are translated to QoS requirements, specified at the level of the interaction model abstractions. We define *gateways* as crucial architecture components, which serve as brokers for both the functional and non-functional (e.g. QoS) properties of the subsystems they hide.

The basic infrastructure is composed of a global wide area network (WAN) that comprises substructures subsumed by the abstraction of a Controller Area Network (CAN). The WAN comprises all that makes the globally available, mostly wired, ostensibly public and wide range network infrastructure. A CAN represents a confined environment in which a certain quality of communication in terms of bandwidth, transmission delays, and reliability can be enforced. The WAN-of-CAN structure allows a hierarchical composition of heterogeneous environments with respect to timeliness: at the lowest level we may find networks with highly predictable communication, controlling physical devices such as sensors and actuators.

This WAN-of-CAN structure is assembled by means of gateways. A gateway is a crucial architectural construct that provides the propagation of QoS constraints on event flows, and on the events proper, while ensuring timeliness confinement between parts of the architecture, namely in what concerns CAN modules. From an architectural point of view, gateways can be seen as artefacts that provide a representation of a certain environment to the outside world. Therefore, they must provide means to specify how this representation will be established and how the events will flow from, and to the outside environment.

CORTEX, in common with many other complex systems, offers a set of basic support services through a middleware layer. Today, the focus of middleware is on interoperability by providing functionally compatible interfaces. A number of mobile-specific distributed systems services have been developed in recent years that aim to operate in challenging mobile environments [4,9,8]. However, to date such services are not designed to offer sufficient levels of dependability or support the highly asynchronous interaction model required by the *CORTEX* computational paradigm. In more detail, when targeting mission and safety critical applications, e.g. traffic management systems, predictability under widely varying load and fault conditions becomes an additional decisive requirement. It is the conflict between technical conditions and the application requirements that makes predictability one of the greatest challenges for the middleware of the future.

The ability to enforce and check timeliness of actions with given coverage assumptions is necessary in order to achieve dependable execution in face of uncertain timeliness. This requires the availability of a number of basic services, such as timing failure detection and clock synchronization. Similarly, discovery services are mandatory, not only in terms of topology, but also in terms of services offered by the infrastructure. Together, these are the distinctive services supplied by the *CORTEX* middleware.

6 APPLICATION SCENARIOS

In this section, we illustrate with a few scenarios the relevance of the *CORTEX* architecture.

6.1 Supporting field workers in the electricity industry

Field workers in the electricity supply industry work in a highly distributed safety critical and real-time environment. Current best working practice is based on a centralised coordination body (the control centre). However, such centralisation inevitably proves to be a bottleneck and potential point of failure during periods of high activity, such as dur-

ing lightning storms. Furthermore, the highly mobile field engineers are often unable to establish contact with the control centre in a timely fashion, although they may well be able to establish ad-hoc dialogues with neighbouring colleagues. By enhancing collaboration between colleagues and replicating the view of the current network state to all field engineers, there is the potential to distribute operational control and co-ordination [6].

One potential application of *CORTEX* involves placing intelligence and monitoring capabilities into the power distribution network infrastructure itself (e.g. at substation switches). This would allow predictable action by providing different levels of dependability in isolated parts of the network. A switch might, for example, take autonomous action to ensure fail-safe behaviour under certain conditions. Alternatively, these 'sentient switches' may take a proactive role in collaborating with field engineers directly. Such collaboration will, we believe, facilitate the establishment of pockets or zones of co-ordination, enabling useful work to be performed, despite the inability to achieve direct communication with a centralised control centre.

6.2 Mountain rescue

Mountain rescue workers are constantly faced with search and rescue operations in which they are called upon to locate stranded, and possibly injured, people in extremely hostile conditions. Such environments also place stringent constraints on mobile computational devices, such as weight, battery life and communications availability. To affect a successful search of a mountain-side requires a co-ordinated effort by a team of rescuers who are themselves vulnerable to hostile weather conditions, can become separated and even injured. In such scenarios, tracking the location of search team members is required in order to provide both the control centre and rescuers with an awareness of the location of those team members involved in the rescue. Sharing of location information poses a technical challenge; the availability of the communications infrastructure and partitioning of the search team is greatly affected by the topology of the mountain terrain and unpredictability of the prevailing weather conditions.

Ad-hoc networking can enable collaborations between neighbouring team members and promote the sharing of information, such as location and medical telemetry, to enhance the effectiveness of a typical search and rescue operation. Moreover, information gathered in the field can be relayed back to remote experts at the base or local accident and emergency departments.

The dynamic nature of such collaborations is poorly supported by existing distributed systems given the general bias towards a reliable and fixed communications infrastructure. The *CORTEX* paradigm fits well to this application domain by supporting the notion of zones that, in this scenario, could represent zones of network availability.

6.3 Next generation cars

As a more futuristic example, consider the reaction of a queue of cars to an accident on a typically busy motorway carriageway. A driver in the queue will be forced to brake suddenly, but when the next driver reacts, she will brake hard enough to stop her car within the remaining braking distance. Each car in the queue reacts similarly. The usual outcome of this behaviour is that a number of drivers will not have sufficient braking distance left and a multiple car collision will occur.

Using the *CORTEX* paradigm, a more co-ordinated approach could be achieved in which vehicles publish events (such as the fact that they are performing an emergency brake or that the car will be stationary in a number of milliseconds) to other interested parties (e.g., cars following within a certain distance). Sentient objects (located within other cars in the queue) can ask to receive the braking event, and when notified can take appropriate braking action (publishing their own braking events). In this way, the entire queue of vehicles can be brought to a halt in a progressive and controlled manner.

7 CONCLUSIONS

In this paper, we have explored the issues that arise in supporting an emerging class of applications that operate independently of direct human control. We have presented the necessary support mechanisms in terms of system architecture and middleware models. Furthermore, we have derived a set of key characteristics for such a model, including sentience, autonomy, large scale, geographical dispersion, mobility and evolution.

The paper introduces a programming model that provides the necessary means for application developers to construct *proactive applications* that employ mobile *sentient objects*. We described the fundamental aspects of the *CORTEX* programming model, proposing anonymous event-based communication and the possibility of specifying constraints and guarantees in the form of QoS properties. We also introduced a number of important concepts for the programming model, e.g. *filters* and *zones*.

The interaction among the objects in the system, which comprise the aspects of communication and co-ordination, are dealt with in the proposed interaction model. We also propose the fundamentals of a system architecture, which defines the mechanisms that are necessary to implement the communication abstractions identified in the interaction model. A key aspect of this architecture is that it must be capable of handling the extremely heterogeneous nature of possible components.

The *CORTEX* project has already taken the first steps towards defining the fundamental paradigms and solutions needed to address this class of sentient and proactive applications. However, as we continue to revise our model, we are constructing several proof-of-concept prototypes to illustrate the feasibility of our ideas. We have been using several application scenarios to drive our work, of which the ones presented in this paper are just an illustrative subset. We expect to publish further results of our ongoing work in a near future.

REFERENCES

[1] Bakre, A. and Badrinath, B.R., M-RPC: A Remote Procedure Call Service for Mobile Clients. Technical Report WINLAB TR-98, Department of Computer Science, Rutgers University, USA, June 1995.

[2] Campbell, A. and Coulson, G., A QoS adaptive transport system: Design, implementation and experience. In *Proceedings of the Fourth ACM Multimedia Conference,* pages117-128, New York, NY, USA, Nov 1996.

[3] Carriero, N. and Gelernter, D., Linda in Context. *Communications of the ACM*, 32(4):444-458, Apr. 1989.

[4] Cheverst, K., Development of a Group Service to Support Collaborative Mobile Groupware. Ph.D. Thesis, Computing Department, Lancaster University, Bailrigg, Lancaster, LA1 4YR, U.K., Apr. 1999.

[5] Cristian, F. and Fetzer, C., The Timed Asynchronous System Model. In *Proceedings of the 28th Annual International Symposium on Fault-Tolerant Computing*, pages 140-149, Munich, Germany, June 1998.

[6] Davies, N. and Friday, A. and Blair, G.S. and Cheverst, K., Distributed Systems Support for Adaptive Mobile Applications. In *ACM Mobile Networks and Applications*, special issue *Mobile Computing – System Services*, 4(5), 1996.

[7] Estrin, D. and Govindan, R. and Heidemann, J., Scalable coordination in sensor networks. In *Proc. of the 5th ACM/IEEE International Conference on Mobile Computing and Networking,* Seattle, WA, USA, 1999.

[8] Franz, W. and Hartenstein, H. and Bochow, B., Internet on the Road via Inter-Vehicle Communications. In *Proc. GI/OCG Annual Conference: Workshop on Mobile Communications over Wireless LAN: Research and Applications,* Vienna, Sept. 2001.

[9] Friday, A., Infrastructure Support for Adaptive Mobile Applications. Ph.D. Thesis, Computing Department, Lancaster University, Bailrigg, Lancaster, LA1 4YR, U.K., Sept. 1996.

[10] Haahr, M. and Cunningham, R. and Cahill. V., Supporting CORBA Applications in a Mobile Environment. In *Proc. of the 5th ACM/IEEE International Conference on Mobile Computing and* Seattle, WA, USA, 1999.

[11] Horstmann, M. and Kirtland, M., DCOM Architecture. http://www.microsoft.com/jini/specs/.

[12] Jensen, E. and Northcutt, J., Alpha: A non-proprietary os for large, complex, distributed real-time. In *Procs. of the IEEE Workshop on Experimental Distributed Systems*, pages 35-41, Alabama, USA, 1990.

[13] JINI Technology 1.1. Specification, Sun Microsystems, http://www.sun.com/jini/specs/.

[14] Joseph, A. and deLespinasse, A. and Tauber, J. and Gifford, D. and Kaashoek, M.F., Rover: A Toolkit for Mobile Information Access. In *Proc. 15th ACM Symp. on Operating System Principles,* Vol.29, pp.156-171, Copper Mountain Resort, Colorado, USA, Dec 1995.

[15] Kopetz, H. and Grünsteidl, G., TTP - A Time-Triggered Protocol for Fault-Tolerant Real-Time Systems.Research Report 12/92, Institut für Technische Informatik, Technische Universität Wien, 1992.

[16] Kopetz, H., Real-Time Systems, Design Principles for Distributed Embedded Applications. Kluwer Academic Publishers, 1997.

[17] Kulik, J. and Rabiner, W. and Balakrishnan, H., Adaptive protocols for information dissemination in wireless sensor networks. In *Proc. of the 5th ACM/IEEE International Conference on Mobile Computing and* Seattle, WA, USA, 1999.

[18] Kümmel, S. and Schill, A. and Volkmann, G., RPC over Advanced Network Technologies: Evaluation and Experiences. In *Proceedings of the 3rd International Workshop on Services in Distributed Networked Environments,* Macau, China, June 1996.

[19] Livani, M.A. and Kaiser, J. and Jia, W.J., Scheduling Hard and Soft Real-Time Communication in the Controller Area Network (CAN). In *23rd IFAC/IFIP Workshop on Real Time Programming*, Shantou, China, June 1998.

[20] Maffeis, S., iBus - The Java Intranet Software Bus. Olsen&Associates, www.olsen.ch, 1997.

[21] Object Management Group. The Common Object Request Broker: Architecture and Specification. *OMG Document 96-03-04*, July 1995.

[22] O'Connell, K. and Dinneen, T. and Collins, S. and Tangney, B. and Harris, N. and Cahill, V., Techniques for Handling Scale and Distribution in Virtual Worlds. In *Proceedings of the 7th ACM SIGOPS European Workshop*, pp.17-24, Connemara, Ireland, Sept. 1996.

[23] Oki, B. and Pfluegl, M. and Seigel, A. and Skeen, D., The information Bus®- An Architecture for Extensible Distributed Systems. *14th ACM Symp. on Operating System Principles*, pp.58-68, Asheville, NC, Dec. 1993.

[24] Rajkumar, R. and Gagliardi, M. and Sha, L., The Real-Time Publisher/Subscribe Inter-Process Communication Model for Distributed Real-Time Systems: Design and Implementation. In *IEEE Real-time Technology and Applications Symposium*, June 1995.

[25] Rufino, J. and Veríssimo, P. and Almeida, C. and Rodrigues, L., Fault-Tolerant Broadcasts in CAN. In *Digest of Papers, The 28th International Symposium on Fault-Tolerant Computing Systems*, Munich, Germany, June 1998.

[26] Starovic, G. and Cahill, V. and Tangney, B., An Event Based Object Model for Distributed Programming. In *OOIS (Object-Oriented Information Systems) '95*, pages 72-86, Dec. 1995.

[27] Tindell, K. and Burns, A., Guaranteed Message latencies for Distributed Safety-Critical Hard Real Time Control Networks. Technical Report YCS229, Dept. of Comp. Science, University of York, May 1994.

[28] Veríssimo, P. and Almeida, C., Quasi-synchronism: a step away from the traditional fault-tolerant real-time system models, *Bulletin of the Technical Committee on Operating Systems and Application Environments (TCOS),* 7(4):35-39, Winter 1995.

[29] Veríssimo, P. and Barrett, P. and Bond, P. and Hilborne, A. and Rodrigues L. and Seaton, D., The Extra Performance Architecture (XPA). In *Delta-4 - A Generic Architecture for Dependable Distributed Computing*, D. Powell ed., pages 211-266, Springer Verlag, ESPRIT Research Reports Series, 1991.

[30] Veríssimo, P. and Casimiro, A. and Fetzer, C., The Timely Computing Base: Timely actions in the presence of uncertain timeliness. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 533-542, New York, USA, June 2000.

[31] Veríssimo, P. and Rodrigues, L., Distributed Systems for System Architects, Kluwer Academic Publishers, 2001.

[32] Zuberi, K.M. and Shin, K.G., Non-Preemptive Scheduling of messages on Controller Area Network for Real-Time Control Applications. Technical Report, University of Michigan, 1995.