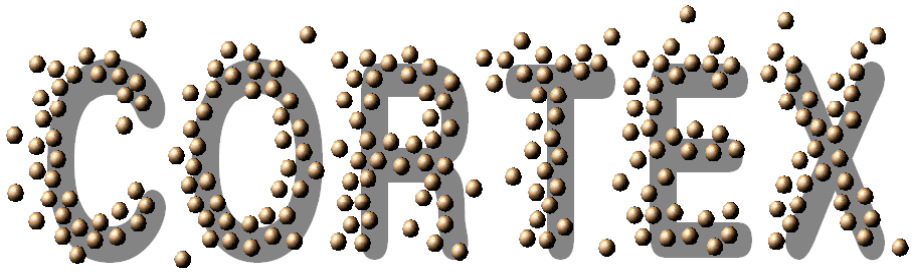
Project IST-2000-26031

CO-operating Real-time senTient objects: architecture and EXperimental evaluation



Final Definition of CORTEX System Architecture

CORTEX Deliverable D11

Version 1.0

July 21, 2004

Revisions

Rev.	Date	Comment
0.1	08/07/2004	Draft document for internal review
1.0	21/07/2004	Final document

Editor

António Casimiro, University of Lisboa

Contributors

Cristiano Brudna, University of Ulm
Vinny Cahill, Trinity College Dublin
António Casimiro, University of Lisboa
Raymond Cunningham, Trinity College Dublin
Jörg Kaiser, University of Ulm
Pedro Martins, University of Lisboa
René Meier, Trinity College Dublin
Paulo Sousa, University of Lisboa
Paulo Veríssimo, University of Lisboa

Address

Faculdade de Ciências da Universidade de Lisboa
Bloco C6, Campo Grande
1749-016 Lisboa
Portugal

Contents

1 Executive Summary and Overview

The objective of work package WP3 is to define the system architecture of CORTEX, focusing on the components that are necessary to implement the communication abstractions developed in WP2. From an architectural point of view it is possible to consider two logical scopes, with different implications in terms of the problems to be dealt. In one hand there is a local scope, over which it makes sense to address issues such as the CORTEX node topology, the definition of local modules to support the required functionality, the problem of interfacing "application" objects, and the provision of runtime support. One the other hand, there is a global scope, where the WAN-of-CAN structure envisaged in CORTEX is of fundamental importance, and where there is room to address issues like wireless operation, and interconnection and composition of CORTEX nodes.

The present deliverable defines the CORTEX architecture in its two scopes: the global one, where the WAN-of-CAN structure plays a fundamental role, and the local one, encompassing the definition of a internal structure of CORTEX nodes.

Section 2 presents two papers where the architectural issues at the CAN level are addressed. The first paper provides a generic architecture that seamlessly integrates real-world and system events and focuses on the specific architectural solution for the event layer, materialized by the COSMIC middleware. The second paper addresses real-time event-based communication when considering the CORTEX WAN-of-CAN structure. In particular, the paper focuses on the quality aspects of communication, presenting abstractions to encapsulate network properties and make them assessable on the level of the event system.

Section 3 focusses on wireless infrastructures as a key technology to implement many of the properties of applications foreseen in CORTEX. The aspects of predictable and reliable behavior, which are particularly relevant in the CORTEX project, are dealt special attention in this section, with the description of a new medium access protocol that specifically takes them into account.

Finally, Section 4 describes the Timely Computing Base, which can be seen as a special architectural component serving the whole system and providing crucial time related services. These services can be used, for instance, to define protocols or middleware components that implement some of the abstractions defined in WP2.

2 CAN Level

2.1 Introduction

During the project CORTEX, it turned out that the general architectural model of WANs-of-CANs is very useful to structure the system on the network level. Therefore, we used this model without substantial changes throughout the entire project. The WAN-of-CANs model constitutes an adequate abstraction to reason about the quality of event dissemination defined by a certain physical network. The WANs-of-CANs architecture was introduced in CORTEX deliverable D4 and the preliminary network infrastructure has been presented.

As elaborated during the project, higher level abstractions are needed to describe the recursive construction of larger quality of service zones and provide a scope for event dissemination. These aspects are covered by two recent publications which are provided in Sections 2.2 and 2.3. Casimiro et al. describes the component oriented system construction and an adequate architecture for a sentient system capable to handle real-world events and system events in a single abstraction. In Meier et al. we present an architectural view on the abstraction layers of heterogeneous network. The abstract layers provide certain functionality to improve the basic network properties with respect addressing, routing and quality of service. The paper discusses the problems of mapping the event layer to the underlying network.

Regarding the infrastructure, the application programming interface (API), and the specific protocol structure, considerable changes have been made compared to the description in D4. Because this relates to basic services and protocols these changes in the API have been treated in deliverable D5 (Preliminary Specification of Basic Services and Protocols). Actual updates of the infrastructure are now moved to deliverable D12 (Final Specification of Basic Services and Protocols).

2.2 An Architectural Framework and a Middleware for Cooperating Smart Components *

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

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

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

ABSTRACT

In a future networked physical world, a myriad of smart sensors and actuators assess and control aspects of their environments and autonomously act in response to it. Examples range in telematics, traffic management, team robotics or home automation to name a few. To a large extent, such systems operate proactively and independently of direct human control driven by the perception of the environment and the ability to organize respective computations dynamically. The challenging characteristics of these applications include sentience and autonomy of components, issues of responsiveness and safety criticality, geographical dispersion, mobility and evolution. A crucial design decision is the choice of the appropriate abstractions and interaction mechanisms. Looking to the basic building blocks of such systems we may find components which comprise mechanical components, hardware and software and a network interface, thus these components have different characteristics compared to pure software components. They are able to spontaneously disseminate information in response to events observed in the physical environment or to events received from other component via the network interface. Larger autonomous components may be composed recursively from these building blocks.

The paper describes an architectural framework and a middleware supporting a component-based system and an integrated view on events-based communication comprising the real world events and the events generated in the system. It starts by an outline of the component-based system construction. The generic event architecture GEAR is introduced which describes the event-based interaction between the components via a generic event layer. The generic event layer hides the different communication channels including the interactions through the environment. An appropriate middleware is presented which reflects these needs and allows to specify events which have quality attributes to express temporal constraints. This is complemented by the notion of event channels which are abstractions of the underlying network and allow to enforce quality attributes. They are established prior to interaction to reserve the needed computational and network resources for highly predictable event dissemination.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures; C.2.4 []: Distributed Systems—*Distributed applications*; C.3 [**Special-Purpose and Application-Based Systems**]: *Real-Time and embedded systems*

General Terms

Design

Keywords

Events-based communication, sentient computing, component-based systems, middleware architectures

1. INTRODUCTION

In recent years we have seen the continuous improvement of technologies that are relevant for the construction of distributed embedded systems, including trustworthy visual, auditory, and location sensing [11], communication and processing. We believe that in a future networked physical world a new class of applications will emerge, composed of a myriad of smart sensors and actuators to assess and control aspects of their environments and autonomously act in response to it. The anticipated challenging characteristics of these applications include autonomy, responsiveness and safety criticality, large scale, geographical dispersion, mobility and evolution.

In order to deal with these challenges, it is of fundamental importance to use adequate high-level models, abstractions and interaction paradigms. Unfortunately, when facing the specific characteristics of the target systems, the shortcomings of current architectures and middleware interaction paradigms become apparent. Looking to the basic building blocks of such systems we may find components which comprise mechanical parts, hardware, software and a network interface. However, classical event/object models are usually software oriented and, as such, when trans-

*This work was partially supported by the EC, through project IST-2000-26031 (CORTEX), and by the FCT, through the Large-Scale Informatic Systems Laboratory (LaSIGE) and project POSI/1999/CHS/33996 (DEFEATS).

CF'04, April 14–16, 2004, Ischia, Italy.

ported to a real-time, embedded systems setting, their harmony is cluttered by the conflict between, on the one side, send/receive of "software" events (message-based), and on the other side, input/output of "hardware" or "real-world" events, register-based. In terms of interaction paradigms, and although the use of event-based models appears to be a convenient solution [10, 22], these often lack the appropriate support for non-functional requirements like reliability, timeliness or security.

This paper describes an architectural framework and a middleware, supporting a component-based system and an integrated view on event-based communication comprising the real world events and the events generated in the system.

When choosing the appropriate interaction paradigm it is of fundamental importance to address the challenging issues of the envisaged sentient applications. Unlike classical approaches that confine the possible interactions to the application boundaries, i.e. to its components, we consider that the environment surrounding the application also plays a relevant role in this respect. Therefore, the paper starts by clarifying several issues concerning our view of the system, about the interactions that may take place and about the information flows. This view is complemented by providing an outline of the component-based system construction and, in particular, by showing that it is possible to compose larger applications from basic components, following an hierarchical composition approach.

This provides the necessary background to introduce the **Generic-Events Architecture (GEAR)**, which describes the event-based interaction between the components via a generic event layer while allowing the seamless integration of physical and computer information flows. In fact, the generic event layer hides the different communication channels, including the interactions through the environment.

Additionally, the event layer abstraction is also adequate for the proper handling of the non-functional requirements, namely reliability and timeliness, which are particularly stringent in real-time settings. The paper devotes particular attention to this issue by discussing the temporal aspects of interactions and the needs for predictability.

An appropriate middleware is presented which reflects these needs and allows to specify events which have quality attributes to express temporal constraints. This is complemented by the notion of *Event Channels (EC)*, which are abstractions of the underlying network while being abstracted by the event layer. In fact, event channels play a fundamental role in securing the functional and non-functional (e.g. reliability and timeliness) properties of the envisaged applications, that is, in allowing the enforcement of quality attributes. They are established prior to interaction to reserve the needed computational and network resources for highly predictable event dissemination.

The paper is organized as follows. In Section 3 we introduce the fundamental notions and abstractions that we adopt in this work to describe the interactions taking place in the system. Then, in Section 4, we describe the component-based approach that allows composition of objects. GEAR is then described in Section 5 and Section 6 focuses on temporal aspects of the interactions. Section 7 describes the COSMIC middleware, which may be used to specify the interaction between sentient objects. A simple example to highlight the ideas presented in the paper appears in Section 8 and Section 9 concludes the paper.

2. RELATED WORK

Our work considers a wired physical world in which a very large number of autonomous components cooperate. It is inspired by many research efforts in very different areas. Event-based systems in general have been introduced to meet the requirements of applications in which entities spontaneously generate information and disseminate it [1, 25, 22]. Intended for large systems and requiring quite complex infrastructures, these event systems do not consider stringent quality aspects like timeliness and dependability issues. Secondly, they are not created to support inter-operability between tiny smart devices with substantial resource constraints.

In [10] a real-time event system for CORBA has been introduced. The events are routed via a central event server which provides scheduling functions to support the real-time requirements. Such a central component is not available in an infrastructure envisaged in our system architecture and the developed middleware TAO (The Ace Orb) is quite complex and unsuitable to be directly integrated in smart devices.

There are efforts to implement CORBA for control networks, tailored to connect sensor and actuator components [15, 19]. They are targeted for the CAN-Bus [9], a popular network developed for the automotive industry. However, in these approaches the support for timeliness or dependability issues does not exist or is only very limited.

A new scheme to integrate smart devices in a CORBA environment is proposed in [17] and has lead to the proposal of a standard by the Object Management Group (OMG) [26]. Smart transducers are organized in clusters that are connected to a CORBA system by a gateway.

The clusters form isolated subnetworks. A special master node enforces the temporal properties in the cluster subnet. A CORBA gateway allows to access sensor data and write actuator data by means of an interface file system (IFS). The basic structure is similar to the WAN-of-CANs structure which has been introduced in the CORTEX project [4]. Islands of tight control may be realized by a control network and cooperate via wired or wireless networks covering a large number of these subnetworks. However, in contrast to the event channel model introduced in this paper, all communication inside a cluster relies on a single technical solution of a synchronous communication channel. Secondly, although the temporal behaviour of a single cluster is rigorously defined, no model to specify temporal properties for cluster-to-CORBA or cluster-to-cluster interactions is provided.

3. INFORMATION FLOW AND INTERACTION MODEL

In this paper we consider a component-based system model that incorporates previous work developed in the context of the IST CORTEX project [5]. As mentioned above, a fundamental idea underlying the approach is that applications can be composed of a large number of smart components that are able to sense their surrounding environment and interact with it. These components are referred to as *sentient objects*, a metaphor elaborated in CORTEX and inspired on the generic concept of *sentient computing* introduced in [12]. Sentient objects accept input events from a variety of different sources (including sensors, but not constrained to that), process them, and produce output events, whereby

they actuate on the environment and/or interact with other objects. Therefore, the following kinds of interactions can take place in the system:

Environment-to-object interactions: correspond to a flow of information from the environment to application objects, reporting about the state of the former, and/or notifying about events taking place therein.

Object-to-object interactions: correspond to a flow of information among sentient objects, serving two purposes. The first is related with complementing the assessment of each individual object about the state of the surrounding space. 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: correspond to a flow of information from an object to the environment, with the purpose of forcing a change in the state of the latter.

Before continuing, we need to clarify a few issues with respect to these possible forms of interaction. We consider that the environment can be a producer or consumer of information while interacting with sentient objects. The environment is the real (physical) world surrounding an object, not necessarily close to the object or limited to certain boundaries. Quite clearly, the information produced by the environment corresponds to the physical representation of real-time entities, of which typical examples include temperature, distance or the state of a door. On the other hand, actuation on the environment implies the manipulation of these real-time entities, like increasing the temperature (applying more heat), changing the distance (applying some movement) or changing the state of the door (closing or opening it). The required transformations between system representations of these real-time entities and their physical representations is accomplished, generically, by sensors and actuators. We further consider that there may exist *dumb* sensors and actuators, which interact with the objects by disseminating or capturing raw transducer information, and *smart* sensors and actuators, with enhanced processing capabilities, capable of "speaking" some more elaborate "event dialect" (see Sections 5 and 6.1). Interaction with the environment is therefore done through sensors and actuators, which may, or may not be part of sentient objects, as discussed in Section 4.2.

State or state changes in the environment are considered as events, captured by sensors (in the environment or within sentient objects) and further disseminated to other potentially interested sentient objects in the system. In consequence, it is quite natural to base the communication and interaction among sentient objects and with the environment on an event-based communication model. Moreover, typical properties of event-based models, such as anonymous and non-blocking communication, are highly desirable in systems where sentient objects can be mobile and where interactions are naturally very dynamic.

A distinguishing aspect of our work from many of the existing approaches, is that we consider that sentient objects may indirectly communicate with each other 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. In other words, when a sentient object actuates on the environment it will be able to observe the state changes in the environment by means of events captured by the sensors. Clearly, other objects might as well capture the same events, thus establishing the above-mentioned indirect communication path.

In systems that involve interactions with the environment it is very important to consider the possibility of communication through the environment. It has been shown that the hidden channels developing through the latter (e.g., feedback loops) may hinder software-based algorithms ignoring them [30]. Therefore, any solution to the problem requires the definition of convenient abstractions and appropriate architectural constructs.

On the other hand, in order to deal with the information flow through the whole computer system and environment in a seamless way, handling "software" and "hardware" events uniformly, it is also necessary to find adequate abstractions. As discussed in Section 5, the Generic-Events Architecture introduces the concept of *Generic Event* and an *Event Layer* abstraction which aim at dealing, among others, with these issues.

4. SENTIENT OBJECT COMPOSITION

In this section we analyze the most relevant issues related with the sentient object paradigm and the construction of systems composed of sentient objects.

4.1 Component-based System Construction

Sentient objects can take several different forms: they can simply be software-based components, but they can also comprise mechanical and/or hardware parts, amongst which the very sensorial apparatus that substantiates "sentience", mixed with software components to accomplish their task. We refine this notion by considering a sentient object as an encapsulating entity, a component with internal logic and active processing elements, able to receive, transform and produce new events. This interface hides the internal hardware/software structure of the object, which may be complex, and shields the system from the low-level functional and temporal details of controlling a specific sensor or actuator.

Furthermore, given the inherent complexity of the envisaged applications, the number of simultaneous input events and the internal size of sentient objects may become too large and difficult to handle. Therefore, it should be possible to consider the hierarchical composition of sentient objects so that the application logic can be separated across as few or as many of these objects as necessary. On the other hand, composition of sentient objects should normally be constrained by the actual hardware component's structure, preventing the possibility of arbitrarily composing sentient objects. This is illustrated in Figure 1, where a sentient object is internally composed of a few other sentient objects, each of them consuming and producing events, some of which only internally propagated.

Observing the figure, and recalling our previous discussion about the possible interactions, we identify all of them here: an object-to-environment interaction occurs between the object controlling a WLAN transmitter and some WLAN receiver in the environment; an environment-to-object interaction takes place when the object responsible for the GPS

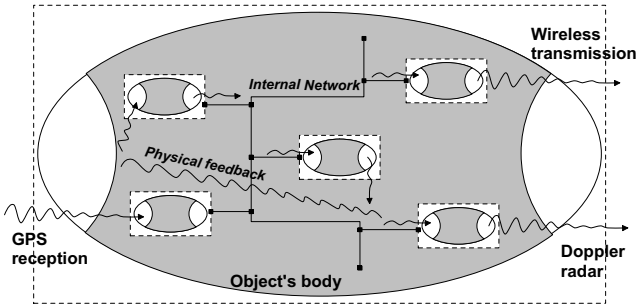


Figure 1: Component-aware sentient object composition.

signal reception uses the information transmitted by the satellites; finally, explicit object-to-object interactions occur internally to the container object, through an internal communication network. Additionally, it is interesting to observe that implicit communication can also occur, whether the physical feedback develops through the environment internal to the container object (as depicted) or through the environment external to this object. However, there is a subtle difference between both cases. While in the former the feedback can only be perceived by objects internal to the container, bounding the extent to which consistency must be ensured, such bounds do not exist in the latter. In fact, the notion of sentient object as an encapsulating entity may serve other purposes (e.g., the confinement of feedback and of the propagation of events), beyond the mere hierarchical composition of objects.

To give a more concrete example of such *component-aware* object composition we consider a scenario of cooperating robots. Each robot is made of several components, corresponding, for instance, to axis and manipulator controllers. Together with the control software, each of these controllers may be a sentient object. On the other hand, a robot itself is a sentient object, composed of the objects materialized by the controllers, and the environment internal to its own structure, or *body*.

This means that it should be possible to define cooperation activities using the events produced by robot sentient objects, without the need to know the internal structure of robots, or the events produced by body objects or by smart sensors within the body. From an engineering point of view, however, this also means that robot sentient object may have to generate new events that reflect its internal state, which requires the definition of a gateway to make the bridge between the internal and external environments.

4.2 Encapsulation and Scoping

Now an important question is about how to represent and disseminate events in a large scale networked world. As we have seen above, any event generated by a sentient object could, in principle, be visible anywhere in the system and thus received by any other sentient object. However, there are substantial obstacles to such universal interactions, originating from the components heterogeneity in such a large-scale setting.

Firstly, the components may have severe performance constraints, particularly because we want to integrate smart sensors and actuators in such an architecture. Secondly, the bandwidth of the participating networks may vary largely. Such networks may be low power, low bandwidth fieldbuses, or more powerful wireless networks as well as high speed backbones. Thirdly, the networks may have widely different reliability and timeliness characteristics. Consider a platoon of cooperating vehicles. Inside a vehicle there may be a field-bus like CAN [8, 9], TTP/A [17] or LIN [20], with a comparatively low bandwidth. On the other hand, the vehicles are communicating with others in the platoon via a direct wireless link. Finally, there may be multiple platoons of vehicles which are coordinated by an additional wireless network layer.

At the abstraction level of sentient objects, such heterogeneity is reflected by the notion of *body-vs-environment*. At the network level, we assume the *WAN-of-CANs* structure [27] to model the different networks. The notion of body and environment is derived from the recursively defined component-based object model. A body is similar to a cell membrane and represents a quality of service container for the sentient objects inside. On the network level, it may be associated with the components coupled by a certain CAN. A CAN defines the dissemination quality which can be expected by the cooperating objects.

In the above example, a vehicle may be a sentient object, whose body is composed of the respective lower level objects (sensors and actuators) which are connected by the internal network (see Figure 1). Correspondingly, the platoon can be seen itself as an object composed of a collection of cooperating vehicles, its body being the environment encapsulated by the platoon zone. At the network level, the wireless network represents the respective CAN. However, several platoons united by their CANs may interact with each other and objects further away, through some wider-range, possible fixed networking substrate, hence the concept of WAN-of-CANs.

The notions of body-environment and WAN-of-CANs are very useful when defining interaction properties across such boundaries. Their introduction obeyed to our belief that a single mechanism to provide quality measures for interactions is not appropriate. Instead, a high level construct for interaction across boundaries is needed which allows to specify the quality of dissemination and exploits the knowledge about body and environment to assess the feasibility of quality constraints. As we will see in the following section, the notion of an *event channel* represents this construct in our architecture. It disseminates events and allows the network independent specification of quality attributes. These attributes must be mapped to the respective properties of the underlying network structure.

5. A GENERIC EVENTS ARCHITECTURE

In order to successfully apply event-based object-oriented models, addressing the challenges enumerated in the introduction of this paper, it is necessary to use adequate architectural constructs, which allow the enforcement of fundamental properties such as timeliness or reliability.

We propose the **Generic-Events Architecture (GEAR)**, depicted in Figure 2, which we briefly describe in what follows (for a more detailed description please refer to [29]). The L-shaped structure is crucial to ensure some of the properties described.

Environment: The physical surroundings, remote and close, solid and etherial, of sentient objects.

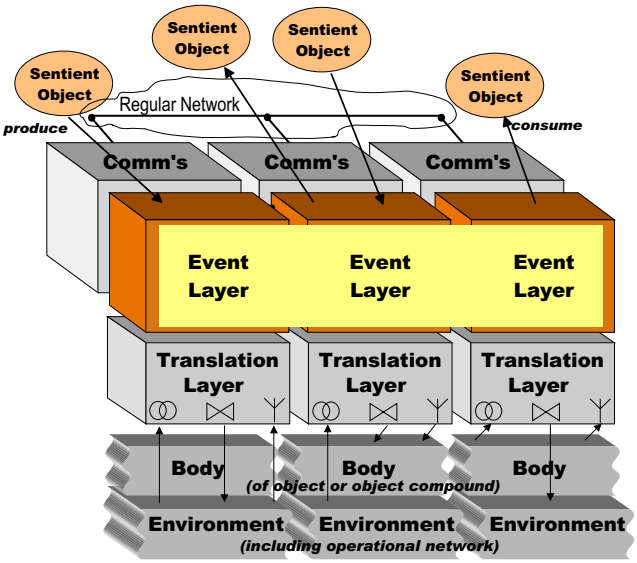


Figure 2: Generic-Events architecture.

Body: The physical embodiment of a sentient object (e.g., the hardware where a mechatronic controller resides, the physical structure of a car). Note that due to the compositional approach taken in our model, part of what is "environment" to a smaller object seen individually, becomes "body" for a larger, containing object. In fact, the body is the "internal environment" of the object. This architecture layering allows composition to take place seamlessly, in what concerns information flow.

Inside a body there may also be implicit knowledge, which can be exploited to make interaction more efficient, like the knowledge about the number of cooperating entities, the existence of a specific communication network or the simple fact that all components are co-located and thus the respective events do not need to specify location in their context attributes. Such intrinsic information is not available outside a body and, therefore, more explicit information has to be carried by an event.

Translation Layer: The layer responsible for physical event transformation from/to their native form to event channel dialect, between environment/body and an event channel. Essentially one doing observation and actuation operations on the lower side, and doing transactions of event descriptions on the other. On the lower side this layer may also interact with dumb sensors or actuators, therefore "talking" the language of the specific device. These interactions are done through *operational networks* (hence the antenna symbol in the figure).

Event Layer: The layer responsible for event propagation in the whole system, through several *Event Channels (EC):*. In concrete terms, this layer is a kind of middleware that provides important event-processing services which are crucial for any realistic event-based system. For example, some of the services that imply the processing of events may include publishing, subscribing, discrimination (zoning, filtering, fusion, tracing), and queuing.

Communication Layer: The layer responsible for "wrapping" events (as a matter of fact, event descriptions in EC dialect) into "carrier" *event-messages*, to be transported to remote places. For example, a sensing event generated by a smart sensor is wrapped in an event-message and disseminated, to be caught by whoever is concerned. The same holds for an actuation event produced by a sentient object, to be delivered to a remote smart actuator. Likewise, this may apply to an event-message from one sentient object to another. Dumb sensors and actuators do not send event-messages, since they are unable to understand the EC dialect (they do not have an event layer neither a communication layer— they communicate, if needed, through operational networks).

Regular Network: This is represented in the horizontal axis of the block diagram by the communication layer, which encompasses the usual LAN, TCP/IP, and real-time protocols, desirably augmented with reliable and/or ordered broadcast and other protocols.

The GEAR introduces some innovative ideas in distributed systems architecture. While serving an object model based on production and consumption of generic events, it treats events produced by several sources (environment, body, objects) in a homogeneous way. This is possible due to the use of a common basic dialect for talking about events and due to the existence of the translation layer, which performs the necessary translation between the physical representation of a real-time entity and the EC compliant format. Crucial to the architecture is the event layer, which uses event channels to propagate events through regular network infrastructures. The event layer is realized by the COSMIC middleware, as described in Section 7.

5.1 Information Flow in GEAR

The flow of information (external environment and computational part) is seamlessly supported by the L-shaped architecture. It occurs in a number of different ways, which demonstrates the expressiveness of the model with regard to the necessary forms of information encountered in real-time cooperative and embedded systems.

Smart sensors produce events which report on the environment. Body sensors produce events which report on the body. They are disseminated by the local event layer module, on an event channel (EC) propagated through the regular network, to any relevant remote event layer modules where entities showed an interest on them, normally, sentient objects attached to the respective local event layer modules.

Sentient objects consume events they are interested in, process them, and produce other events. Some of these events are destined to other sentient objects. They are published on an EC using the same EC dialect that serves, e.g., sensor originated events. However, these events are semantically of a kind such that they are to be subscribed by the relevant sentient objects, for example, the sentient objects composing a robot controller system, or, at a higher level, the sentient objects composing the actual robots in

a cooperative application. Smart actuators, on the other hand, merely consume events produced by sentient objects, whereby they accept and execute actuation commands. Alternatively to "talking" to other sentient objects, sentient objects can produce events of a lower level, for example, actuation commands on the body or environment. They publish these exactly the same way: on an event channel through the local event layer representative. Now, if these commands are of concern to local actuator units (e.g., body, including internal operational networks), they are passed on to the local translation layer. If they are of concern to a remote smart actuator, they are disseminated through the distributed event layer, to reach the former. In any case, if they are also of interest to other entities, such as other sentient objects that wish to be informed of the actuation command, then they are also disseminated through the EC to these sentient objects.

A key advantage of this architecture is that event-messages and physical events can be globally ordered, if necessary, since they all pass through the event layer. The model also offers opportunities to solve a long lasting problem in real-time, computer control, and embedded systems: the inconsistency between message passing and the feedback loop information flow subsystems.

6. TEMPORAL ASPECTS OF THE INTERACTIONS

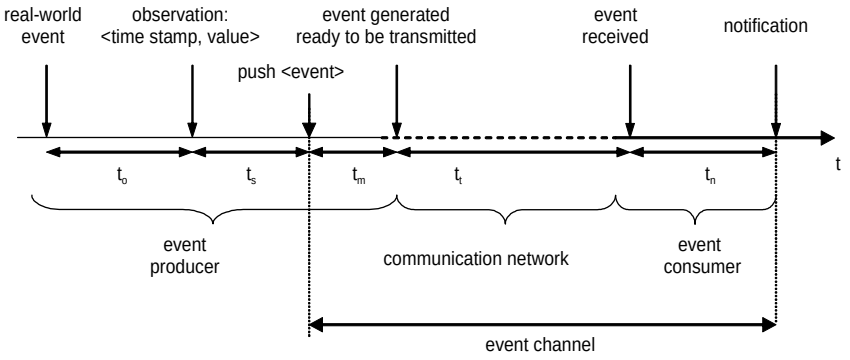
Any interaction needs some form of predictability. If safety critical scenarios are considered as it is done in CORTEX, temporal aspects become crucial and have to be made explicit. The problem is how to define temporal constraints and how to enforce them by appropriate resource usage in a dynamic ad-hoc environment. In an system where interactions are spontaneous, it may be also necessary to determine temporal properties dynamically. To do this, the respective temporal information must be stated explicitly and available during run-time. Secondly, it is not always ensured that temporal properties can be fulfilled. In these cases, adaptations and timing failure notification must be provided [2, 28]. In most real-time systems, the notion of a deadline is the prevailing scheme to express and enforce timeliness. However, a deadline only weakly reflect the temporal characteristics of the information which is handled. Secondly, a deadline often includes implicit knowledge about the system and the relations between activities. In a rather well defined, closed environment, it is possible to make such implicit assumptions and map these to execution times and deadlines. E.g. the engineer knows how long a vehicle position can be used before the vehicle movement outdates this information. Thus he maps this dependency between speed and position on a deadline which then assures that the position error can be assumed to be bounded. In a open environment, this implicit mapping is not possible any more because, as an obvious reason, the relation between speed and position, and thus the error bound, cannot easily be reverse engineered from a deadline. Therefore, our event model includes explicit quality attributes which allow to specify the temporal attributes for every individual event. This is of course an overhead compared to the use of implicit knowledge, but in a dynamic environment such information is needed.

To illustrate the problem, consider the example of the position of a vehicle. A position is a typical example for $\langle$*time, value*$\rangle$ entity [30]. Thus, the position is useful if we can determine an error bound which is related to time, e.g. if we want a position error below 10 meters to establish a safety property between cooperating cars moving with 5 m/sec, the position has a validity time of 2 seconds. In a $\langle$*time, value*$\rangle$ entity entity we can trade time against the precision of the value. This is known as value over time and time over value [18]. Once having established the time-value relation and captured in event attributes, subscribers of this event can locally decide about the usefulness of an information. In the GEAR architecture temporal validity is used to reason about safety properties in a event-based system [29]. We will briefly review the respective notions and see how they are exploited in our COSMIC event middleware.

Consider the timeline of generating an event representing some real-time entity [18] from its occurrence to the notification of a certain sentient object (Figure 3). The real-time entity is captured at the sensor interface of the system and has to be transformed in a form which can be treated by a computer. During the time interval t_0 the sensor reads the real-time entity and a time stamp is associated with the respective value. The derived $\langle$*time, value*$\rangle$ entity represents an observation. It may be necessary to perform substantial local computations to derive application relevant information from the raw sensor data. However, it should be noted that the time stamp of the observation is associated with the capture time and thus independent from further signal processing and event generation. This close relationship between capture time and the associated value is supported by smart sensors described above.

The processed sensor information is assembled in an event data structure after t_s to be published to an event channel. As is described later, the event includes the time stamp of generation and the temporal validity as attributes.

The temporal validity is an application defined measure for the expiration of a $\langle$*time, value*$\rangle$. As we explained in the example of a position above, it may vary dependent on application parameters. Temporal validity is a more general concept than that of a deadline. It is independent of a certain technical implementation of a system. While deadlines may be used to schedule the respective steps in an event generation and dissemination, a temporal validity is an intrinsic property of a $\langle$*time, value*$\rangle$ entity carried in an event. A temporal validity allows to reason about the usefulness of information and is beneficial even in systems in which timely dissemination of events cannot be enforced because it enables timing failure detection at the event consumer. It is obvious that deadlines or periods can be derived from the temporal validity of an event. To set a deadline, knowledge of an implementation, worst case execution times or message dissemination latencies is necessary. Thus, in the timeline of Figure 3 every interval may have a deadline. Event dissemination through soft real-time channels in COSMIC exploits the temporal validity to define dissemination deadlines. Quality attributes can be defined, for instance, in terms of $\langle$*validity interval, omission degree*$\rangle$ pairs. These allow to characterize the usefulness of the event for a certain application, in a certain context. Because of that, quality attributes of an event clearly depend on higher level issues, such as the nature of the sentient object or of the smart sensor that produced the event. For instance, an event containing an indication of some vehicle speed must have different quality attributes depending on the kind of vehicle



t_o : time to obtain an observation
t_s : time to process sensor reading
t_m : time to assemble an event message
t_t : time to transfer the event on the regular network
t_n : time for notification on the consumer site

Figure 3: Event processing and dissemination.

from which it originated, or depending on its current speed. The same happens with the position event of the car example above, whose validity depends on the current speed and on a predefined required precision. However, since quality attributes are strictly related with the semantics of the application or, at least, with some high level knowledge of the purpose of the system (from which the validity of the information can be derived), the definition of these quality attributes may be done by exploiting the information provided at the programming interface. Therefore, it is important to understand how the system programmer can specify non-functional requirements at the API, and how these requirements translate into quality attributes assigned to events. While temporal validity is identified as an intrinsic event property, which is exploited to decide on the usefulness of data at a certain point in time, it is still necessary to provide a communication facility which can disseminate the event before the validity is expired.

In a WAN-of-CANs network structure we have to cope with very different network characteristics and quality of service properties. Therefore, when crossing the network boundaries the quality of service guarantees available in a certain network will be lost and it will be very hard, costly and perhaps impossible to achieve these properties in the next larger area of the WAN-of CANs structure. CORTEX has a couple of abstractions to cope with this situation (network zones, body/environment) which have been discussed above. From the temporal point of view we need a high level abstraction like the temporal validity for the individual event now to express our quality requirements of the dissemination over the network. The ⟨*bound, coverage*⟩ pair, introduced in relation with the TCB [28] seems to be an appropriate approach. It considers the inherent uncertainty of networks and allows to trade the quality of dissemination against the resources which are needed. In relation with the event channel model discussed later, the ⟨*bound, coverage*⟩ pair allows to specify the quality properties of an event channel independently of specific technical issues. Given the typical environments in which sentient applications will operate, where it is difficult or even impossible to provide timeliness or reliability guarantees, we proposed an alternative way to handle non-functional application requirements, in relation with the TCB approach [28]. The proposed approach exploits intrinsic characteristics of applications, such as fail-safety, or time-elasticity, in order to secure QoS specifications of the form ⟨*bound, coverage*⟩. Instead of constructing systems that rely on guaranteed bounds, the idea is to use (possibly changing) bounds that are secured with a constant probability all over the execution. This obviously requires an application to be able to adapt to changing conditions (and/or changing bounds) or, if this is not possible, to be able to perform some safety procedures when the operational conditions degrade to an unbearable level. The bounds we mentioned above refer essentially to timeliness bounds associated to the execution of local or distributed activities, or combinations thereof. From these bounds it is then possible to derive the quality attributes, in particular validity intervals, that characterize the events published in the event channel.

6.1 The Role of Smart Sensors and Actuators

Smart devices encapsulate hardware, software and mechanical components and provide information and a set of well specified functions and which are closely related to the interaction with the environment. The built-in computational components and the network interface enable the implementation of a well-defined high level interface that does not just provide raw transducer data, but a processed, application-related set of events. Moreover, they exhibit an autonomous spontaneous behaviour. They differ from general purpose nodes because they are dedicated to a certain functionality which complies to their sensing and actuating capabilities while general purpose node may execute any program.

Concerning the sentient object model, smart sensors and actuators may be basic sentient objects themselves, consuming events from the real-world environment and producing the respective generic events for the system's event layer or,

vice versa consuming a generic event and converting it to a real-world event by an actuation. Smart components therefore constitute the periphery, i.e. the real-world interface of a more complex sentient object. The model of sentient objects also constitutes the framework to built more complex "virtual" sensors by relating multiple (primary, i.e. sensors which directly sense a physical entity) sensors.

Smart components translate events of the environment to an appropriate form available at the event layer or, vice versa, transform a system event into an actuation. For smart components we can assume that:

- Smart components have dedicated resources to perform a specific function.
- These resources are not used for other purposes during normal real-time operation.
- No local temporal conflicts occur that will change the observable temporal behaviour.
- The functions of a component can usually only be changed during a configuration procedure which is not performed when the component is involved in critical operations.
- An observation of the environment as a ⟨time,value⟩ pair can be obtained with a bounded jitter in time.

Many predictability and scheduling problems arise from the fact, that very low level timing behaviours have to be handled on a single processor. Here, temporal encapsulation of activities is difficult because of the possible side effects when sharing a single processor resource. Consider the control of a simple IR-range detector which is used for obstacle avoidance. Dependent on its range and the speed of a vehicle, it has to be polled to prevent the vehicle from crashing into an obstacle. On a single central processor, this critical activity has to be coordinated with many similar, possibly less critical functions. It means that a very fine grained schedule has to be derived based purely on the artifacts of the low level device control. In a smart sensor component, all this low level timing behaviour can be optimized and encapsulated. Thus we can assume temporal encapsulation similar to information hiding in the functional domain. Of course, there is still the problem to guarantee that an event will be disseminated and recognized in due time by the respective system components, but this relates to application related events rather than the low artifacts of a device timing. The main responsibility to provide timeliness guarantees is shifted to the event layer where these events are disseminated. Smart sensors thus lead to network centric system model. The network constitute the shared resource which has to be scheduled in a predictable way. The COSMIC middleware introduced in the next section is an approach to provide predictable event dissemination for a network of smart sensors and actuators.

7. AN EVENT MODEL AND MIDDLEWARE FOR COOPERATING SMART DEVICES

An event model and a middleware suitable for smart components must support timely and reliable communication and also must be resource efficient. **COSMIC (COoperating Smart devices)** is aimed at supporting the interaction between those components according to the concepts introduced so far. Based on the model of a WAN-of-CANs, we assume that the components are connected to some form of CAN as a fieldbus or a special wireless sensor network which provides specific network properties. E.g. a fieldbus developed for control applications usually includes mechanisms for predictable communication while other networks only support a best effort dissemination. A gateway connects these CANs to the next level in the network hierarchy. The event system should allow the dynamic interaction over a hierarchy of such networks and comply with the overall CORTEX generic event model. Events are typed information carriers and are disseminated in a publisher/ subscriber style [24, 7], which is particularly suitable because it supports generative, anonymous communication [3] and does not create any artificial control dependencies between producers of information and the consumers. This decoupling in space (no references or names of senders or receivers are needed for communication) and the flow decoupling (no control transfer occurs with a data transfer) are well known [24, 7, 14] and crucial properties to maintain autonomy of components and dynamic interactions.

It is obvious that not all networks can provide the same QoS guarantees and secondly, applications may have widely differing requirements for event dissemination. Additionally, when striving for predictability, resources have to be reserved and data structures must be set up before communication takes place. Thus, these things can not predictably be made on the fly while disseminating an event. Therefore, we introduced the notion of an event channel to cope with differing properties and requirements and have an object to which we can assign resources and reservations. The concept of an event channel is not new [10, 25], however, it has not yet been used to reflect the properties of the underlying heterogeneous communication networks and mechanisms as described by the GEAR architecture. Rather, existing event middleware allows to specify the priorities or deadlines of events handled in an event server. Event channels allow to specify the communication properties on the level of the event system in a fine grained way. An event channel is defined by:

$$event\_channel := \langle subject, quality\_attributeList, handlers \rangle$$

The subject determines the types of events event which may be issued to the channel. The quality attributes model the properties of the underlying communication network and dissemination scheme. These attributes include latency specifications, dissemination constraints and reliability parameters. The notion of zones which represent a guaranteed quality of service in a subnetwork support this approach. Our goal is to handle the temporal specifications as ⟨*bound, coverage*⟩ pairs [28] orthogonal to the more technical questions of how to achieve a certain synchrony property of the dissemination infrastructure. Currently, we support quality attributes of event channels in a CAN-Bus environment represented by explicit synchrony classes.

The COSMIC middleware maps the channel properties to lower level protocols of the regular network. Based on our previous work on predictable protocols for the CAN-Bus, COSMIC defines an abstract network which provides hard, soft and non real-time message classes [21].

Correspondingly, we distinguish three event channel classes according to their synchrony properties: hard real-time channels, soft real-time channels and non-real-time channels.

Hard real-time channels (HRTC) guarantee event propagation within the defined time constraints in the presence

of a specified number of omission faults. HRTECs are supported by a reservation scheme which is similar to the scheme used in time-triggered protocols like TTP [16][31], TTP/A [17], and TTCAN [8]. However, a substantial advantage over a TDMA scheme is that due to CAN-Bus properties, bandwidth which was reserved but is not needed by a HRTEC can be used by less critical traffic [21].

Soft real-time channels (SRTC) exploit the temporal validity interval of events to derive deadlines for scheduling. The validity interval defines the point in time after which an event becomes temporally inconsistent. Therefore, in a real-time system an event is useless after this point and may me discarded. The transmission deadline (DL) is defined as the latest point in time when a message has to be transmitted and is specified in a time interval which is derived from the expiration time:

$$t_{event\_ready} < DL < t_{expiration} - \Delta_{notification}$$

$t_{expiration}$ defines the point in time when the temporal validity expires. $\Delta_{notification}$ is the expected end-to-end latency which includes the transfer time over the network and the time the event may be delayed by the local event handling in the nodes. As said before, event deadlines are used to schedule the dissemination by SRTECs. However, deadlines may be missed in transient overload situations or due to arbitrary arrival times of events. On the publisher side the application's exception handler is called whenever the event deadline expires before event transmission. At this point in time the event is also not expected to arrive at the subscriber side before the validity expires. Therefore, the event is removed from the sending queue. On the subscriber side the expiration time is used to schedule the delivery of the event. If the event cannot be delivered until its expiration time it is removed from the respective queues allocated by the COSMIC middleware. This prevents the communication system to be loaded by outdated messages.

Non-real-time channels do not assume any temporal specification and disseminate events in a best effort manner. An instance of an event channel is created locally, whenever a publisher makes an announcement for publication or a subscriber subscribes for an event notification. When a publisher announces publication, the respective data structures of an event channel are created by the middleware. When a subscriber subscribes to an event channel, it may specify context attributes of an event which are used to filter events locally. E.g. a subscriber may only be interested in events generated at a certain location. Additionally the subscriber specifies quality properties of the event channel. A more detailed description of the event channels can be found in [13].

Currently, COSMIC handles all event channels which disseminate events beyond the CAN network boundary as non real-time event channels. This is mainly because we use the TCP/IP protocol to disseminate events over wireless links or to the standard Ethernet. However, there are a number of possible improvements which can easily be integrated in the event channel model. The Timely Computing Base (TCB) [28] can be exploited for timing failure detection and thus would provide awareness for event dissemination in environments where timely delivery of events cannot be enforced. Additionally, there are wireless protocols which can provide timely and reliable message delivery [6, 23] which may be exploited for the respective event channel classes.

Events are the information carriers which are exchanged between sentient objects through event channels. To cope with the requirements of an ad-hoc environment, an event includes the description of the context in which it has been generated and quality attributes defining requirements for dissemination. This is particularly important in an open, dynamic environment where an event may travel over multiple networks. An event instance is specified as:

$$event := \langle subject, context\_attributeList, quality\_attributeList, contents \rangle$$

A subject defines the type of the event and is related to the event contents. It supports anonymous communication and is used to route an event. The subject has to match to the subject of the event channel through which the event is disseminated. Attributes are complementary to the event contents. They describe individual functional and non-functional properties of the event. The context attributes describe the environment in which the event has been generated, e.g. a location, an operational mode or a time of occurrence. The quality attributes specify timeliness and dependability aspects in terms of ⟨*validity interval, omission degree*⟩ pairs. The validity interval defines the point in time after which an event becomes temporally inconsistent [18]. As described above, the temporal validity can be mapped to a deadline. However, usually a deadline is an engineering artefact which is used for scheduling while the temporal validity is a general property of a ⟨*time, value*⟩ entity. In a environment where a deadline cannot be enforced, a consumer of an event eventually must decide whether the event still is temporally consistent, i.e. represents a valid ⟨*time, value*⟩ entity.

7.1 The Architecture of the COSMIC Middleware

On the architectural level, COSMIC distinguish three layers roughly depicted in Figure 4. Two of them, the event layer and the abstract network layer are implemented by the COSMIC middleware. The event layer provides the API for the application and realizes the abstraction of event and event channels.

The abstract network implements real-time message classes and adapts the quality requirements to the underlying real network. An event channel handler resides in every node. It supports the programming interface and provides the necessary data structures for event-based communication. Whenever an object subscribes to a channel or a publisher announces a channel, the event channel handler is involved. It initiates the binding of the channel's subject, which is represented by a network independent unique identifier to an address of the underlying abstract network to enable communication [14]. The event channel handler then tightly cooperates with the respective handlers of the abstract network layer to disseminate events or receive event notifications. It should be noted that the QoS properties of the event layer in general depend on what the abstract network layer can provide. Thus, it may not always be possible to e.g. support hard real-time event channels because the abstract network layer cannot provide the respective guarantees. In [13], we describe the protocols and services of the abstract network layer particularly for the CAN-Bus.

As can be seen in Figure 4, the hard real-time (HRT) message class is supported by a dedicated handler which is able to provide the time triggered message dissemination.

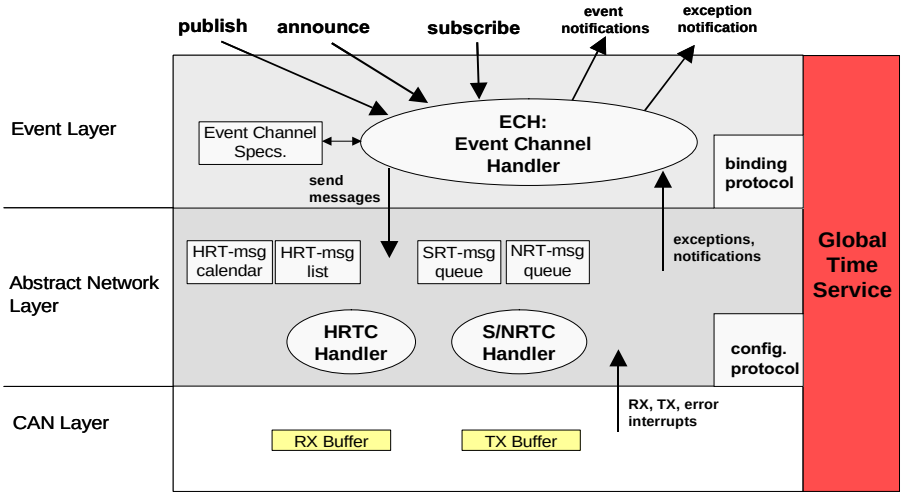


Figure 4: Architecture layers of COSMIC.

The HRT handler maintains the HRT message list, which contains an entry for each local HRT message to be sent. The entry holds the parameters for the message, the activation status and the binding information. Messages are scheduled on the bus according to the HRT message calendar which comprises the precise start time for each time slot allocated for a message. Soft real-time message queues order outgoing messages according to their transmission deadlines derived from the temporal validity interval. If the transmission deadline is exceeded, the event message is purged out of the queue. The respective application is notified via the exception notification interface and can take actions like trying to publish the event again or publish it to a channel of another class. Incoming event messages are ordered according to their temporal validity. If an event message arrive, the respective applications are notified. At the moment, an outdated message is deleted from the queue and if the queue runs out of space, the oldest message is discarded. However, there are other policies possible depending on event attributes and available memory space. Non real-time messages are FIFO ordered in a fixed size circular buffer.

7.2 Status of COSMIC

The goal for developing COSMIC was to provide a platform to seamlessly integrate smart tiny components in a large system. Therefore, COSMIC should run also on the small, resource constraint devices which are built around 16-Bit or even 8-Bit micro-controllers. The distributed COSMIC middleware has been implemented and tested on various platforms. Under RT-Linux, we support the real-time channels over the CAN Bus as described above. The RT-Linux version runs on Pentium processors and is currently evaluated before we intent to port it to a smart sensor or actuator. For the interoperability in a WAN-of-CANs environment, we only provide non real-time channels at the moment. This version includes a gateway between the CAN-bus and a TCP/IP network. It allows us to use a standard wireless 802.11 network. The non real-time version of COSMIC is available on Linux, RT-Linux and on the micro-controller families C167 (Infineon) and 68HC908 (Motorola). Both micro-controllers have an on-board CAN controller and thus do not require additional hardware components for the network. The memory footprint of COSMIC is about 13 Kbyte on a C167 and slightly more on the 68HC908 where it fits into the on-board flash memory without problems. Because only a few channels are required on such a smart sensor or actuator component, the requirement of RAM (which is a scarce resource on many single chip systems) to hold the dynamic data structures of a channel is low. The COSMIC middleware makes it very easy to include new smart sensors in an existing system. Particularly, the application running on a smart sensor to condition and process the raw physical data must not be aware of any low level network specific details. It seamlessly interacts with other components of the system exclusively via event channels.

The demo example, briefly described in the next chapter, is using a distributed infrastructure of tiny smart sensors and actuators directly cooperating via event channels over heterogeneous networks.

8. AN ILLUSTRATIVE EXAMPLE

A simple example for many important properties of the proposed system showing the coordination through the environment and events disseminated over the network is the demo of two cooperating robots depicted in Figure 5.

Each robot is equipped with smart distance sensors, speed sensors, acceleration sensors and one of the robots (the “guide” (KURT2) in front (Figure 5)) has a tracking camera allowing to follow a white line. The robots form a WAN-of-CANs system in which their local CANs are interconnected via a wireless 802.11 network. COSMIC provides the event layer for seamless interaction. The “blind” robot (N.N.) is searching the guide randomly. Whenever the blind robot detects (by its front distance sensors) an obstacle, it checks whether this may be the guide. For this purpose, it dynamically subscribes to the event channel disseminating distance events from rear distance sensors of the guide(s) and compares these with the distance events from its local front sensors. If the distance is approximately the same it infers that it is really behind a guide. Now N.N. also subscribes to the event channels of the tracking camera and the speed sensors

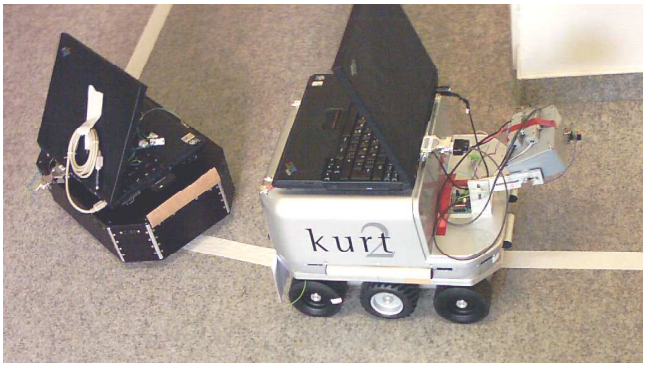


Figure 5: Cooperating robots.

to follow the guide. The demo application highlights the following properties of the system:

1. Dynamic interaction of robots which is not known in advance. In principle, any two a priori unknown robots can cooperate. All what publishers and subscribers have to know to dynamically interact in this environment is the subject of the respective event class. A problem will be to receive only the events of the robot which is closest. A robot identity does not help much to solve this problem. Rather, the position of the event generation entity which is captured in the respective attributes can be evaluated to filter the relevant event out of the event stream. A suitable wireless protocol which uses proximity to filter events has been proposed by Meier and Cahill [22] in the CORTEX project.

2. Interaction through the environment. The cooperation between the robots is controlled by sensing the distance between the robots. If the guide detects that the distance grows, it slows down. Respectively, if the blind robot comes too close it reduces its speed. The local distance sensors produce events which are disseminated through a low latency, highly predictable event channel. The respective reaction time can be calculated as function of the speed and the distance of the robots and define a dynamic dissemination deadline for events. Thus, the interaction through the environment will secure the safety properties of the application, i.e. the follower may not crash into the guide and the guide may not loose the follower. Additionally, the robots have remote subscriptions to the respective distance events which are used to check it with the local sensor readings to validate that they really follow the guide which they detect with their local sensors. Because there may be longer latencies and omissions, this check occasionally will not be possible. The unavailability of the remote events will decrease the quality of interaction and probably and slow down the robots, but will not affect safety properties.

3. Cooperative sensing. The blind robot subscribes to the events of the line tracking camera. Thus it can "see" through the eye of the guide. Because it knows the distance to the guide and the speed as well, it can foresee the necessary movements. The proposed system provides the architectural framework for such a cooperation. The respective sentient object controlling the actuation of the robot receives as input the position and orientation of the white line to be tracked. In the case of the guide robot, this information is directly delivered as a body event with a low latency and a high reliability over the internal network. For the follower robot, the information comes also via an event channel but with different quality attributes. These quality attributes are reflected in the event channel description. The sentient object controlling the actuation of the follower is aware of the increased latency and higher probability of omission.

9. CONCLUSION AND FUTURE WORK

The paper addresses problems of building large distributed systems interacting with the physical environment and being composed from a huge number of smart components. We cannot assume that the network architecture in such a system is homogeneous. Rather multiple "edge-" networks are fused to a hierarchical, heterogeneous wide area network. They connect the tiny sensors and actuators perceiving the environment and providing sentience to the application. Additionally, mobility and dynamic deployment of components require the dynamic interaction without fixed, a priori known addressing and routing schemes. The work presented in the paper is a contribution towards the seamless interaction in such an environment which should not be restricted by technical obstacles. Rather it should be possible to control the flow of information by explicitly specifying functional and temporal dissemination constraints.

The paper presented the general model of a sentient object to describe composition, encapsulation and interaction in such an environment and developed the Generic Event Architecture GEAR which integrates the interaction through the environment and the network. While appropriate abstractions and interaction models can hide the functional heterogeneity of the networks, it is impossible to hide the quality differences. Therefore, one of the main concerns is to define temporal properties in such an open infrastructure. The notion of an event channel has been introduced which allows to specify quality aspects explicitly. They can be verified at subscription and define a boundary for event dissemination. The COSMIC middleware is a first attempt to put these concepts into operation. COSMIC allows the interoperability of tiny components over multiple network boundaries and supports the definition of different real-time event channel classes.

There are many open questions that emerged from our work. One direction of future research will be the inclusion of real-world communication channels established between sensors and actuators in the temporal analysis and the ordering of such events in a cause-effect chain. Additionally, the provision of timing failure detection for the adaptation of interactions will be in the focus of our research. To reduce network traffic and only disseminate those events to the subscribers which they are really interested in and which have a chance to arrive timely, the encapsulation and scoping schemes have to be transformed into respective multi-level filtering rules. The event attributes which describe aspects of the context and temporal constraints for the dissemination will be exploited for this purpose. Finally, it is intended to integrate the results in the COSMIC middleware to enable experimental assessment.

10. REFERENCES

[1] J. Bacon, K. Moody, J. Bates, R. Hayton, C. Ma, A. McNeil, O. Seidel, and M. Spiteri. Generic support for distributed applications. *IEEE Computer*, 33(3):68–76, 2000.

[2] L. B. Becker, M. Gergeleit, S. Schemmer, and E. Nett. Using a flexible real-time scheduling strategy in a distributed embedded application. In *Proc. of the 9th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Lisbon, Portugal, Sept. 2003.

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

[4] A. Casimiro (Ed.). Preliminary definition of cortex system architecture. CORTEX project, IST-2000-26031, Deliverable D4, Apr. 2002.

[5] CORTEX project Annex 1, Description of Work. Technical report, CORTEX project, IST-2000-26031, Oct. 2000. http://cortex.di.fc.ul.pt.

[6] R. Cunningham and V. Cahill. Time bounded medium access control for ad hoc networks. In *Proceedings of the Second ACM International Workshop on Principles of Mobile Computing (POMC'02)*, pages 1–8, Toulouse, France, Oct. 2002. ACM Press.

[7] P. T. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. Technical Report DSC ID:200104, EPFL, Lausanne, Switzerland, 2001.

[8] T. Führer, B. Müller, W. Dieterle, F. Hartwich, R. Hugel, and M.Walther. Time triggered communication on CAN, 2000. http://www.can-cia.org/can/ttcan/fuehrer.pdf.

[9] R. B. GmbH. CAN Specification Version 2.0. Technical report, Sept. 1991.

[10] T. Harrison, D. Levine, and D. Schmidt. The design and performance of a real-time corba event service. In *Proceedings of the 1997 Conference on Object Oriented Programming Systems, Languages and Applications (OOPSLA)*, pages 184–200, Atlanta, Georgia, USA, 1997. ACM Press.

[11] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, aug 2001.

[12] A. Hopper. The Clifford Paterson Lecture, 1999 Sentient Computing. *Philosophical Transactions of the Royal Society London*, 358(1773):2349–2358, Aug. 2000.

[13] J. Kaiser, C. Mitidieri, C. Brudna, and C. Pereira. COSMIC: A Middleware for Event-Based Interaction on CAN. In *Proc. 2003 IEEE Conference on Emerging Technologies and Factory Automation*, Lisbon, Portugal, Sept. 2003.

[14] J. Kaiser and M. Mock. Implementing the real-time publisher/subscriber model on the controller area network (CAN). In *Proceedings of the 2nd International Symposium on Object-oriented Real-time distributed Computing (ISORC99)*, Saint-Malo, France, May 1999.

[15] K. Kim, G. Jeon, S. Hong, T. Kim, and S. Kim. Integrating subscription-based and connection-oriented communications into the embedded CORBA for the CAN Bus. In *Proceedings of the IEEE Real-time Technology and Application Symposium*, May 2000.

[16] H. Kopetz and G. Grünsteidl. TTP - A Time-Triggered Protocol for Fault-Tolerant Real-Time Systems. Technical Report rr-12-92, Institut für Technische Informatik, Technische Universität Wien, Treilstr. 3/182/1, A-1040 Vienna, Austria, 1992.

[17] H. Kopetz, M. Holzmann, and W. Elmenreich. A Universal Smart Transducer Interface: TTP/A. *International Journal of Computer System, Science Engineering*, 16(2), Mar. 2001.

[18] H. Kopetz and P. Veríssimo. Real-time and Dependability Concepts. In S. J. Mullender, editor, *Distributed Systems, 2nd Edition*, ACM-Press, chapter 16, pages 411–446. Addison-Wesley, 1993.

[19] S. Lankes, A. Jabs, and T. Bemmerl. Integration of a CAN-based connection-oriented communication model into Real-Time CORBA. In *Workshop on Parallel and Distributed Real-Time Systems*, Nice, France, Apr. 2003.

[20] Local Interconnect Network: LIN Specification Package Revision 1.2. Technical report, Nov. 2000.

[21] M. Livani, J. Kaiser, and W. Jia. Scheduling hard and soft real-time communication in the controller area network. *Control Engineering*, 7(12):1515–1523, 1999.

[22] R. Meier and V. Cahill. Steam: Event-based middleware for wireless ad hoc networks. In *Proceedings of the International Workshop on Distributed Event-Based Systems (ICDCS/DEBS'02)*, pages 639–644, Vienna, Austria, 2002.

[23] E. Nett and S. Schemmer. Reliable real-time communication in cooperative mobile applications. *IEEE Transactions on Computers*, 52(2):166–180, Feb. 2003.

[24] B. Oki, M. Pfluegl, A. Seigel, and D. Skeen. The information bus - an architecture for extensible distributed systems. *Operating Systems Review*, 27(5):58–68, 1993.

[25] O. M. G. (OMG). CORBAservices: Common Object Services Specification - Notification Service Specification, Version 1.0, 2000.

[26] O. M. G. (OMG). Smart transducer interface, initial submission, June 2001.

[27] P. Veríssimo, V. Cahill, A. Casimiro, K. Cheverst, A. Friday, and J. Kaiser. Cortex: Towards supporting autonomous and cooperating sentient entities. In *Proceedings of European Wireless 2002*, Florence, Italy, Feb. 2002.

[28] P. Veríssimo and A. Casimiro. The Timely Computing Base model and architecture. *Transactions on Computers - Special Issue on Asynchronous Real-Time Systems*, 51(8):916–930, Aug. 2002.

[29] P. Veríssimo and A. Casimiro. Event-driven support of real-time sentient objects. In *Proceedings of the 8th IEEE International Workshop on Object-oriented Real-time Dependable Systems*, Guadalajara, Mexico, Jan. 2003.

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

2.3 An Event Model for Real-Time Systems in Mobile Environments

René Meier[1], Jörg Kaiser[2], Barbara Hughes[1], Vinny Cahill[1]
[1]Department of Computer Science, Trinity College Dublin, Ireland
[2]Department of Computer Structures, University of Ulm, Germany
{rene.meier, barbara.hughes, vinny.cahill}@cs.tcd.ie, kaiser@informatik.uni-ulm.de

ABSTRACT
This paper describes an event model that has been designed to address the predictability requirements of applications operating in mobile environments based on hierarchically structured WAN-of-CANs network. The event model supports an event channel concept for modeling the guarantees provided by the underlying, heterogeneous communication infrastructure. The networks that comprise such a WAN-of-CANs may provide fundamentally different degrees of quality of service and as a result can be viewed as zones within which certain guarantees can be enforced. Event channels operating in CAN-based subnetworks with typically strong timing behavior may support hard temporal and reliability attributes whereas channels interconnecting these subnetworks using wireless networks support weaker timing attributes.

Keywords
Event-based middleware, timely event delivery, mobile computing, CAN networks, wireless networks.

1. INTRODUCTION

Advances in information technology encourage new classes of applications that are based on a large number of networked components acting autonomously in response to a myriad of sensors and actuators to assess and control aspects of the environment. Examples range in telematics, traffic management or home automation to name a few. To a large extend, such systems operate proactively and independently of direct human control driven by the perception of the environment and the ability to organize respective computations dynamically. The challenging characteristics of these applications include sentience and autonomy of components, issues of responsiveness and safety criticality, geographical dispersion, mobility and evolution.

Such systems require a degree of co-operation, adaptability, extensibility and reliability that is not available today. The problem is to provide a communication and interaction scheme that supports a large-scale many-to-many communication relation typical for these applications. Additionally, it should be possible to seamlessly disseminate relevant information generated by deeply embedded controllers to all interested entities in the global network.

The contribution of the paper is a model for a real-time event system on top of a heterogeneous communication system. Such a system may be composed from networks with largely different quality characteristics ranging from highly predictable real-time busses to ad-hoc wireless networks in which mobility of nodes result in frequent link failures. The focus of the paper is on the quality aspects of communication, explicitly providing a system architecture that reflects a realistic network model in large-scale cooperative applications. It presents abstractions to encapsulate network properties and make them assessable on the level of the event system. Related work on event systems does not allow to model quality properties resulting from heterogeneity explicitly [1-3], or it deals with mechanisms that mask network heterogeneity, thereby neglecting respective real-time and quality issues.

1.1 Overview

The event-based communication model represents a paradigm for middleware that asynchronously interconnects the components that comprise distributed applications [4]. Event-based middleware has been recognized as being well suited to addressing the requirements of mobile application [5-8] and to connecting the components in control applications with timeliness requirements [1-3]. However, to date event models have not been designed to offer sufficient levels of dependability for real-time systems while supporting mobile application components.

This paper presents an event model addressing the predictability requirements of large-scale applications operating in mobile environments based on heterogeneous communication

Conference '00, Month 1-2, 2000, City, State.

infrastructures. Our event model defines an abstract network model and provides mappings for the envisaged network types. These networks may include Controller Area Networks (CAN), Local Area Networks (LAN), and Wide Area Networks (WAN), especially those based on wireless technology.

The underlying communication infrastructure typically consists of a hierarchically structured *WAN-of-CANs network*, an internetwork whose subnetworks will typically be CANs that are interconnected by means of wireless LANs and WANs. The networks that comprise such a WAN-of-CANs typically provide fundamentally different degrees of Quality of Service (QoS), ranging from CAN networks with strong timing and reliability behavior to wireless network supporting mobile application components and as a result providing weaker guarantees. We regard individual networks as QoS containers, called *zones*, within which given degrees of predictability in terms of timing and reliability can be enforced.

Our event model provides a programming model based on the concept of event channels. A number of event channel classes with different temporal and reliability attributes are supported for integrating the real-time aspects into the event channel model. Depending on the guarantees available from the underlying network, these event channels can be mapped to certain QoS zones. Generally, channels can be associated with zones providing equal or stronger guarantees.

In order to address the requirements of real-time systems, we support classes of hard, soft, and non real-time event channels. Hard real-time event channels are considered to meet all temporal requirements under the specified fault assumptions. Soft real-time event channels are scheduled according to deadlines. Nevertheless, these deadlines are not guaranteed during transient overload conditions. Non real-time event channels are typically used for best-effort event delivery without any specified timeliness requirements. In general, any class of event channel may be mapped to zones representing CAN-based subnetworks whereas zones including wireless network can only support guarantees sufficient for soft and non real-time event channels. However, we envisaged using a predictable medium access protocol, such as the Time-Bounded Medium Access Control (TBMAC) [9] protocol, in order to provide soft real-time deadlines with a high probability for event channels propagating events in wireless networks.

1.2 Paper Organization

The remainder of this paper is structured as follows: Section 2 presents the system architecture including the underlying network infrastructure and the classes of event channels. In section 3, we discuss arising issues when applying real-time to a WAN-of-CANs network, especially those based on wireless technology. Section 4 focuses on the programming model supported by our event model and section 5 outlines the communications architecture. Section 6 concludes this paper by summarising our work and outlining the issues that remain open for future work.

2. SYSTEM ARCHITECTURE

When striving for a real-time event system, the communications architecture substantially affects the temporal and reliability properties of event dissemination. Thus, in quality terms the underlying network is not transparent for the application using the event system. The goal for our event system is to express the quality requirements of event dissemination on the abstraction level of events rather than to resort to lower network details. The middleware then automatically maps these requirements to the underlying network. Hence, we will briefly introduce the general assumptions about the network architecture.

2.1 General Network Structure

From an application point of view, the event system should support cooperating autonomous entities, such as cars, robots, and people wearing computers and moving through a smart environment. As an example consider autonomous mobile robots which may be instrumented with all kinds of smart sensors, such as lasers, radars, cameras, navigational sensors, and chemical sensors, to achieve an adequate perception of their environment. The necessary highly reactive behaviour of such an individual robot has to be achieved by a tight cooperation of the sensor/actor system over some special purpose network inside the robot, called a CAN (Controller Area Network in a broad sense). Additionally, the vehicles may communicate via a wireless link carrying out joint tasks. At an abstract layer, this is modelled as islands of tight control cooperating via a WAN-of-CANs structure [10]. We assume a certain guaranteed level of predictability as an intrinsic property of a CAN. A CAN therefore constitutes a zone of coherent QoS provision.

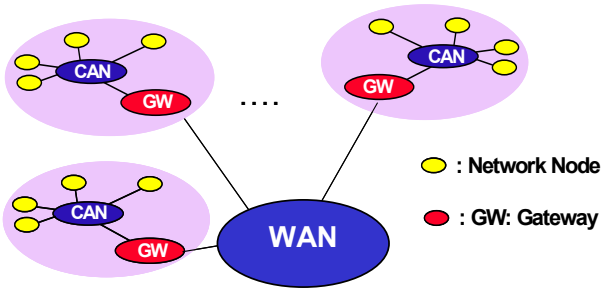


Figure 1. The WAN-of-CANs network architecture.

The general structure depicted in Figure 1 is recursive in a sense that a CAN may be composed from a hierarchy of networks itself. This issue is not further discussed in this paper. It should only be noted that each of the interconnected subnets constitutes a zone in which a certain QoS is defined. The gateways have multiple functions. Firstly, they route events over network boundaries by maintaining an event interface that is transparent with respect to functional properties throughout the network. For non-functional quality attributes they provide the needed awareness about the network properties. Thus, they have to marshal events depending on the underlying abstract network infrastructure, which is described below. Secondly, they constitute filters that allow to

scope events according to their context and their quality attributes as introduced in section 4. A realisation of this structure for cooperating mobile robots internally equipped with a CAN-Bus, which is extended via gateways to a wireless LAN for event dissemination between the robots is presented in [11]. Furthermore, a middleware service that has been designed for interconnecting mobile application components using ad hoc wireless local area networks has been described in [12].

2.2 Abstract Communication Layers

There is a trade-off between the predictability of communication and the needed resources. At the safe end, all communication is statically planned and resources have to be assigned anticipating worst-case load and failure assumptions. However, this may only be required for a small number of highly critical services. In fact, critical system services as could use such highly predictable links to meet its temporal and reliability requirements, e.g. tight sensor/actor control loops e.g. for crash avoidance or motor and brake control. In most cases less critical events have to be accommodated by the communication system, which also allow more dynamic system behaviour. However, also for these events, temporal parameters may be needed, e.g. the specification of when an event should be delivered or how long the event is valid. The different requirements are reflected by event channel classes with different properties. On the architectural level, we distinguish three layers roughly depicted in Figure 2.

On the event layer, the main abstractions are events and different classes of event channels. As described in section 4, this layer enables the application to specify channels with different QoS properties. Mapping the abstractions of the event layer directly to the underlying network is a tough challenge because the usual abstractions on the network layer are low-level messages. Hence, this layer does not match the requirements of group communication, subject-based addressing or the QoS specifications defined for channels.

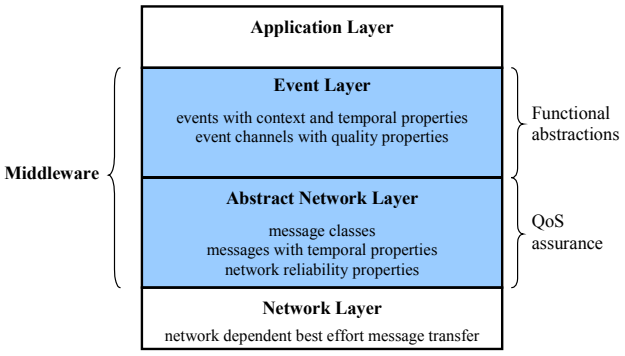


Figure 2. Architectural layer.

Therefore, an abstract network layer is introduced which enriches the properties of the raw network (this may not just be a physical network, but a specific MAC-layer or even a higher OSI level) by additional properties and communication services as e.g. some form of reliable broadcast or group communication and temporal guarantees for the message transfer. This separation of concerns supports modularity and allows for an easier adaptation of the event layer to networks with widely differing characteristics.

It should be noted that the QoS properties of the event layer in general depend on what the abstract network layer can provide. Thus, it may not always be possible to e.g. support highly safety critical hard real-time event channels because the abstract network layer cannot provide the respective guarantees. Therefore, the event channels supported by the event system are dependent on the zones in which the respective protocols can be provided. As an example, [13] describes the protocols and services of the abstract network layer for a particular CAN-Bus.

3. REAL-TIME AND MOBILITY

Previous research in real-time event-based communication [1, 3] has focused on wired networks assuming a static number of location fixed nodes and a known fault hypothesis. Transmission schedules for avoiding collisions are typically planned statically and the correctness of these schedules regarding timing conflicts and temporal overlaps are verified off-line. However, such an approach would be inappropriate for achieving real-time guarantees for wireless environments, especially when using ad hoc networks. The characteristics of wireless networks that render such an approach unsuitable include highly dynamic connections, location-dependent coverage and contention, and its susceptibility to the physical environment.

3.1 Dynamic Connections

Wireless network connections between mobile nodes are typically established dynamically due to the unpredictability of node movements. Maintaining accurate network information in such an environment, which is essential for enforcing delivery guarantees, requires the propagation of updates reflecting the actual network topology. The frequency for disseminating these updates relates directly to the level of node mobility and consequently results in additional, potentially unpredictable communication and computational load.

Link failures are the norm rather that the exception in wireless networks. Links may fail as a result of various reasons including individual node failures, for example due to low battery power, physical obstacles blocking a transmission, and nodes moving out of transmission range. Significantly, link failures may result in network partitions where groups of node operate independently of each other. Adapting to link failures relies on timely and accurate topology updates, which may be affected by the unpredictability of the network and by the presence of partitions. Updates may be confined to certain partitions, which implies partial knowledge of the network topology.

Timeliness guarantees for event delivery rely on a transmission schedule that reflects the priorities and deadlines associated with events and on the precision of such a schedule, which derives from the accuracy of the available typology information. Hence, providing guarantees with a high probability is hard in wireless environments as the required topology information is likely to be out of date or even incomplete. Moreover, it may be impossible to deliver an event to all subscribers in the presence of partitions.

3.2 Location-Dependent Coverage And Contention

Coverage can be defined as the geographical area to which a particular node can send messages using single-hop or multi-hop communication. The area covered by the wireless transmitters of mobile nodes typically changes over time depending on the movements of individual nodes. Consequently, coverage describes the set of nodes, which is likely to change over time, to which a specific node can propagate events at a given time. Kanodia et al. [14] propose a protocol for ordered event scheduling in ad hoc wireless networks. This work focuses on topologies in which all nodes reside within single-hop transmission range of each other and as a result assumes that these nodes have the same information describing the network topology. Tobagi and Kleinrock [15] as well as Kanodia et al. [14] discuss topologies where nodes use multi-hop communication when propagating messages. Such nodes may have different probabilities of accessing a wireless communication channel and different throughput capabilities, depending on properties such as spatial reuse and contention degree. Unequal channel access is detrimental to achieving distributed agreement and results in unpredictable transmission delays, which leads to unbound delivery latency.

Wireless transmissions in ad hoc networks are broadcast through a physical communication channel shared by a group of nodes. This implies that the contention level of a specific channel depends on its location. In order to avoid collisions at a certain location, nodes sense an ongoing transmission and restrain from sending until the channel becomes available. Hence, nodes must exploit a means to arbitrate spatial contentions between the transmissions in their spatial locality. Luo and Lu [16] propose a fair channel access model that ensures coordination among spatially contending transmissions. However, this work assumes that individual transmissions are decoupled from each other and that scheduling can be performed independently for each transmission. This cannot be assumed in applications comprising highly mobile nodes. Moreover, neither Tobagi and Kleinrock [15] nor Luo and Lu [16] address temporal or reliability constraints, which are essential for providing end-to-end delivery guarantees.

3.3 Susceptibility To The Physical Environment

The physical environment has a critical impact on wireless communication. Many physical objects such as vehicles and building as well as people themselves may represent obstacles that effect transmissions. Obstacles impede wireless signals and result in a reduction of the available signal strength. Consequently, they may lead to packet loss as well as to unpredictable transmission delays and jitter. The dynamic aspect of parts of the physical environment causes unpredictable occurrences and durations of such impeding effects. This results in significant fluctuations between intervals of high and low connectivity. Significantly, this unpredictable, dynamic behavior of the physical environment causes variations in the quality of service available from the network.

4. THE PROGRAMMING MODEL

As a consequence of the interaction with the physical world, the pace of interactions and of the computational progress in the distributed system is dictated by the progression of real time and the properties of the environment. To support coordination of actions, cooperation between mobile entities and fusion of elementary events in the sentient object model [17], the events must be disseminated in a predictable way.

An event encapsulates the description of an observation about the change in the environment or the system. Hence, it is related to an occurrence in the real world, e.g. observed by a sensor, or an in the control system itself, e.g. generated by some control program. An event is an instance of an event type, which is characterized by subject, attributes, and content:

event := <subject, attribute_list, content >.

The subject is attached to an event as a tag to explicitly identify the event's type and enable basic filtering and some form of type checking on event dissemination. An event is further characterized by a set of functional and non-functional attributes. Functional attributes are related to the application relevant context in which an event has been generated and may include information like location, range of dissemination (relative location or proximity) and time of occurrence, or system related information like a mode of operation or a network zone in a WAN-of-CANs architecture. Non-functional attributes represent intrinsic properties and quality aspects of an event. These may be deadlines necessary to express temporal requirements of event dissemination or a validity interval according to an aging function. A deadline is used to schedule an event in the communication system and define priority relations to other events. The validity interval defines an application related interval that may go beyond the deadline when the usefulness of the event expires. It allows a certain tolerance of transient overload when the event cannot be delivered within the deadline but eventually to discard an event and purge the system from outdated events. This is particularly important in a real time event system to prevent the already outdated events to still compete for resources with the actual events. Finally, the content of an event carries the data that is represented as a structured set of functional parameters. The content is accessible by specific methods.

Predictability incorporates the timely delivery of events under anticipated load and failure conditions. The nature of events may range from a safety-critical event signaling that a crash of a mobile vehicle with an obstacle is about to happen to the dissemination of non-critical events describing a room temperature or illumination intensity that have less demanding predictability requirements. As a matter of fact, there is a trade-off between the resources expended and the degree of predictability. Consequently, the respective requirements for the underlying abstract network may range from synchronous reliable broadcasts [11, 18] to best effort communication.

To facilitate the specification of the particular requirements for event dissemination the notion of event channels is introduced. An event channel encapsulates properties of the underlying communication system and allows specifying quality attributes on

an abstraction level where it is assessable to an application programmer. The benefits are twofold: Firstly, a check can be performed whether the non-functional attributes of an event match to the quality attributes defined for the event channel. This allows early timing failure detection. Secondly, predictability requires that resources are available when they are needed. An event channel can be established and the necessary resources can be assigned by the middleware before an event has to be disseminated. The general form of an event channel representation is given by:

event_channel := <subject, attribute_list, handlers >

In contrast to the attributes of an event, which describes the properties of a single individual occurrence of an event, the attributes of the event channel abstract the properties of the underlying communication network and dissemination scheme. Therefore the attributes define quality properties and include e.g. latency, dissemination constraints and reliability parameters. The subject of the event channel must match the subject of the event that is disseminated through the channel. The "handlers"-field allows specifying notification and exception handlers for the event channel.

According to the need in most real-time systems, particularly, if we assume the network architecture introduced in section 2, event channels with different timeliness and reliability properties must be supported. We distinguish three event channel classes: Hard Real-Time Event Channels (HRTEC), Soft Real-Time Event Channels (SRTEC) and Non Real-Time Event Channels (NRTEC). A HRTEC offers rigorous guarantees for discrete control based on sporadic events as well as for continuous control requiring periodic events like sensor readings and control feedback. For sporadic events a maximum latency will be guaranteed while for periodic events the goal is to achieve a low period- and latency-jitter. The guarantees are maintained under an anticipated number of omission failures. Events published to a SRTEC are scheduled according to the Earliest Deadline First (EDF) algorithm. As outlined below, deadlines may be missed in situations of transient overload or due to the arbitrary arrival times of messages. Finally, a NRTEC disseminates events that have no timeliness requirements.

The transport of events through a hard real-time event channel (HRTEC) is synchronous and reliable. The properties of a HRTEC are defined by: 1.) a known upper bound for the transport latency, i.e. the interval between the point in time when an event message becomes ready and its delivery; 2.) a known upper bound for the latency jitter, i.e. the variance of the transport latency; 3.) a known upper bound of the period jitter for periodic events, i.e. the variance on the period; 4.) a fault assumption under which the properties 1.)- 3.) are valid. In order to offer such properties, a HRTEC transparently handles redundant transmissions of events and guarantees that the respective publisher has a privileged access to the communication network. A HRTC s is based on a highly predictable abstract network, e.g. enabling the reservation of network resources. Highly predictable and reliable protocols such as [9, 18, 19] can provide the respective quality of service in the abstract network layer.

Soft real-time events have timeliness requirement that are expressed by deadlines and validity intervals. Thus, a SRTC has to reflect these properties. Soft real-time event messages become ready at any time and are scheduled according to their transmission deadlines. Different from HRTCs, SRTECs do not use reservations. The transmission deadline is defined as the latest point in time when a message has to be transmitted and events are scheduled according to an earliest deadline first algorithm. However, because a message can not be interrupted during its transmission and messages may become ready at arbitrary points in time, EDF will not always take the right scheduling decisions (only a clairvoyant scheduler would be able to do so) and situations of temporal conflicts and transient overload may occur. In theses situations, messages will still be transmitted at a later time in a best effort manner. An SRT event message eventually will be discarded if its transmission time is delayed beyond its temporal validity. The expiration time is an application specific parameter, which may be defined according to some value function.

NRTCs are used for events that do not have timeliness requirements. They are primarily intended for configuration and maintenance purposes. While HRTC and SRTC disseminate events of restricted length to meet the responsiveness requirements of real-time systems, NRTC may transfer bulk data.

5. THE COMMUNICATIONS ARCHITECTURE

The design of the communications architecture is motivated by our approach to defining an abstract network model describing the characteristics of the underlying WAN-of-CANs architecture and by our programming model based on our concept of event channels with different timeliness and reliability properties. We employ a mechanism for enforcing hard real-time event channel properties in CAN networks and another mechanism based on bounding the propagation range of events for providing soft real-time guarantees with high probability in wireless networks.

5.1 Mapping Event Channels To Zones

Applications may comprise numerous components representing real-world objects that may be mobile and distributed over a large geographical area. Such components are typically location aware and depending on their location may interact using different parts of the underlying WAN-of-CANs network architecture. Our abstract network model describes the characteristics of such a network infrastructure dividing it into zones reflecting the available quality of the network service. Applications may define various event channels disseminating events with different temporal and reliability properties. Depending on the guarantees available from the underlying network, these event channels can be mapped to certain QoS zones. Generally, event channels can be associated with zones providing equal or stronger guarantees. For example, a traffic management application may include vehicles interacting trough wireless networks in order to exchange information on the current traffic situation thereby contributing to better driver awareness and consequently to safer driving. Such information may include an accident notification disseminated by

a broken-down car to approaching vehicles. Various components representing intra vehicle objects, such as breaks, accelerator, and speedometer, and lights, might communicate using a CAN-based network. Such inter-vehicle and intra-vehicle communication may use different event channel classes for interconnecting their respective components. Hard real-time event channels may be mapped to a zone incorporating intra vehicle components whereas event channels with weaker delivery guarantees may be mapped to a zone comprising components using wireless communication.

Multiple event channels may be mapped to a particular zone sharing the available resources. Such event channels typically disseminate events describing different information and may support different properties. For example, intra-vehicle communication may include a hard real-time event channel disseminating events on behalf of breaks and accelerator and a non real-time event channel controlling the lights of the vehicle.

Event channels may connect components across multiple parts of the network architecture and as a result may be mapped to multiple zones. The properties that can be enforced by such an event channel depend on the level of QoS provided by each of the zones involved. An event channel can be associated with multiple zones if every zone involved provides equal or stronger guarantees. Moreover, an event channel may only operate across the boundaries of multiple zones if the underlying networks are connected through designated gateway components. Such gateways act as producer and consumer of events on either side of the networks they connect. Events received on one side are disseminated on the other side and vice versa. In addition, gateways allow applications to specify network specific mapping of event data and attributes and handle implicit attributes, such as location. A gateway may use a location service to retrieve its own location and may append this location information to events generated by nearby nodes lacking direct access to a location service. For example, a gateway located in a vehicle may connect a CAN-based intra-vehicle network and a wireless inter-vehicle network. It may attach its own location to events generated by nodes on the CAN network as these reside close to the gateway and maintain their location relative to the gateway.

5.2 Mapping Event Channels To CAN Networks

Event channels may support hard real-time guarantees in zones represented by special CAN-networks. We exploited the specific mechanisms of a CAN-Bus, which is popular in the automotive industry, to design an abstract network layer which can support different real-time guarantees for the delivery of messages. The focus of this work has been on using the hardware supported, lower level priority mechanisms of the CAN-Bus to schedule messages according to the suggested real-time classes. The abstract network layer thus implements reliable hard-real time message delivery and also the weaker forms of guarantees. On top of this abstract network layer a publisher/subscriber protocol has been devised [3] for real-time control applications providing all real-time event channel classes introduced earlier. For a detailed description of mapping the real-time channels to this different message classes the reader is referred to [13].

5.3 Mapping Event Channels To Wireless Networks

As we have discussed in section 3, enforcing hard real-time guarantees in wireless environments in general and in ad hoc networks in particular is problematical due to the dynamic nature of these networks. However, we argue that soft real-time event channels can be supported in zones that contain wireless networks. Significantly, we envisage using a set of techniques in order to provide soft real-time guarantees with a high probability.

We propose to bound the propagation range of events in wireless environments [12] by defining event channel attributes that describe geographical areas within which events are valid. Such proximity-based event dissemination represents a natural way to limit the scope of an event channel, thereby allowing entities to interact based on their current location. An example scenario illustrating such behavior might include a broken-down car disseminating an accident notification to vehicles in its vicinity.

Moreover, we envisaged using a predictable medium access protocol, such as the light-weight, location-aware, atomic multicast protocol for Time-Bounded Medium Access Control (TBMAC) [9], when propagating event notifications. The TBMAC protocol is based on time-division multiple access with dynamic but predictable slot allocation and has been designed for use in multi-hop ad hoc networks. It provides, with high probability, time-bounded access to the wireless medium that can be exploited by event channels with guaranteed response time requirements.

6. CONCLUSIONS

We have presented an approach to incorporating the topology of a heterogeneous communication infrastructure into an event-based programming model. We have described how an event model may address the predictability requirements of applications operating in mobile environments based on hierarchically structured WAN-of-CANs network. An abstract network model reflecting the fundamentally different levels of quality of service available from the subnetworks that comprise such a WAN-of-CANs network defines QoS containers that can be viewed as zones within which certain guarantees can be enforced. Our programming model is based on a concept of event channels supporting a number of event channel classes with different temporal and reliability attributes. Depending on the guarantees available from the underlying network, these event channels can be mapped to certain QoS zones. Generally, event channels can be associated with zones providing equal or stronger guarantees.

We have introduced two prototype implementations of our event model. The CAN version enforces hard-real time guarantees in CAN-based subnetworks using lower level mechanisms of the CAN-Bus. The LAN version uses techniques, including geographical bounding of the event propagation range and predictable medium access, to provide soft real-time guarantees with a high probability in wireless networks.

Although we have discussed and addressed some of the fundamental issues arising when applying event-based programming to real-time systems in mobile environments, certain issues remain open for future work. We are currently

investigating means for applications to program entities acting as gateway components. Approaches, for example based on exploiting rule-based programming models, have to be provided for specifying network specific mappings of event data and attributes. Furthermore, our QoS framework needs to be extended in order to incorporate an access control mechanism for proactively managing the number of entities using a specific event channel.

7. ACKNOWLEDGMENTS

The work described in this paper was partly supported by the Irish Higher Education Authority's Programme for Research in Third Level Institutions cycle 0 (1998-2001) and by the FET programme of the Commission of the EU under research contract IST-2000-26031 (CORTEX).

8. REFERENCES

[1] T. Harrison, D. Levine, and D. Schmidt, "The Design and Performance of a Real-Time CORBA Event Service," in *Proceedings of the 1997 Conference on Object- Oriented Programming Systems, Languages and Applications (OOPSLA)*. Atlanta, Georgia, USA: ACM Press, 1997, pp. 184-200.

[2] R. Rajkumar, M. Gagliardi, and L. Sha, "The Real-Time Publisher/Subscriber Inter-Process Communication Model for Distributed Real-Time Systems: Design and Implementation," in *Proceedings of the IEEE Real-time Technology and Applications Symposium*, 1995.

[3] J. Kaiser and M. Mock, "Implementing the Real-Time Publisher/Subscriber Model on the Controller Area Network (CAN)," in *Proceedings of the 2nd International Symposium on Object-oriented Real-time distributed Computing (ISORC99)*. Saint-Malo, France, 1999.

[4] J. Bacon, K. Moody, J. Bates, R. Hayton, C. Ma, A. McNeil, O. Seidel, and M. Spiteri, "Generic Support for Distributed Applications," *IEEE Computer*, vol. 33, pp. 68-76, 2000.

[5] G. Cugola, E. D. Nitto, and A. Fuggetta, "The JEDI Event-Based Infrastructure and its Application to the Development of the OPSS WFMS," *IEEE Transactions on Software Engineering (TSE)*, vol. 27, pp. 827-850, 2001.

[6] Y. Huang and H. Garcia-Molina, "Publish/Subscribe in a Mobile Environment," in *Proceedings of the Second ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDe'01)*. Santa Barbara, CA, USA, 2001, pp. 27-34.

[7] R. Meier, "Communication Paradigms for Mobile Computing," *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R)*, vol. 6, 2002.

[8] H.-A. Jacobsen, "Middleware Services for Selective and Location-based Information Dissemination in Mobile Wireless Networks," presented at Advanced Topic Workshop on Middleware for Mobile Computing (IFIP/ACM Middleware 2001), Heidelberg, Germany, 2001.

[9] R. Cunningham and V. Cahill, "Time Bounded Medium Access Control for Ad Hoc Networks," in *Proceedings of the Second ACM International Workshop on Principles of Mobile Computing (POMC'02)*. Toulouse, France: ACM Press, 2002, pp. 1-8.

[10] P. Verissimo, V. Cahill, A. Casimiro, K. Cheverst, A. Friday, and J. Kaiser, "CORTEX: Towards Supporting Autonomous and Cooperating Sentient Entities," in *Proceedings of the European Wireless Conference*. Florence, Italy, 2002.

[11] J. Kaiser and C. Brudna, "A Publisher/Subscriber Architecture Supporting Interoperability of the CAN-bus and the Internet," in *Proceedings of the IEEE International Workshop on Factory Communication Systems (WFCS 2002)*. Västeras, Sweden, 2002.

[12] R. Meier and V. Cahill, "STEAM: Event-Based Middleware for Wireless Ad Hoc Networks," in *Proceedings of the International Workshop on Distributed Event-Based Systems (ICDCS/DEBS'02)*. Vienna, Austria, 2002, pp. 639-644.

[13] J. Kaiser, C. Brudna, and C. Mitidieri, "A Real-Time Event Channel Model for the CAN Bus," in *Proceedings of the Eleventh International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS 2003)*. Nice, France, 2003.

[14] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly, "Ordered Packet Scheduling in Wireless Ad Hoc Networks: Mechanisms and Performance Analysis," in *Proceedings of ACM MOBIHOC 2002*. Lausanne, Switzerland, 2002.

[15] F. A. Tobagi and L. Kleinrock, "Packet Switching in Radio Channels: Part II - The Hidden Terminal Problem in Carrier Sense Multiple-Access and the Busy-Tone Solution," *IEEE Transactions on Communications*, vol. 23, pp. 1417-1433, 1975.

[16] H. Luo and S. Lu, "A Topology-Independent Fair Queuing Model in Ad Hoc Wireless Networks," in *Proceedings of the IEEE International Conference on Network Protocols (ICNP 2000)*. Osaka, Japan, 2000.

[17] A. Fitzpatrick, G. Biegel, S. Clarke, and V. Cahill, "Towards a Sentient Object Model," presented at Workshop on Engineering Context-Aware Object Oriented Systems and Environments (OOPSLA/ECOOSE'02), Seattle, Washington, USA, 2002.

[18] F. Cristian, "Synchronous Atomic Broadcast for Redundant Broadcast Channels," *The Journal of Real-Time Systems*, vol. 2, pp. 195-212, 1990.

[19] J. Kaiser and M. A. Livani, "Achieving Fault-Tolerant Ordered Broadcasts in CAN," in *Proceedings of the Third European Dependable Computing Conference (EDCC-3)*. Prague, Czech Republic, 1999.

3 Wireless Level

3.1 Introduction

The widespread deployment and use of wireless data communications is generally recognised as being the next major advance in the information technology industry. In the long term, wireless data networks will represent a key enabling technology in the emergence of a new class of mission-critical computer systems as described in [1]. From a wireless communications perspective, key properties of these applications will be:

1. Mobility
2. Autonomy
3. Large scale
4. Time and Safety criticality
5. Geographical dispersion

In the next section, we introduce existing wireless networking technology in terms of these key properties. In the following section we cover existing research with particular reference to ad hoc networks and discuss how this research has yet to address the time and safety criticality properties mentioned above. In section 3.4, we describe a new medium access protocol for ad hoc networks that addresses these time and safety criticality properties that will be evaluated as part of the CORTEX project. This evaluation will also consider how this new protocol supports other key properties such as large scale, geographical dispersion and mobility.

3.2 Wireless Networks

Wireless networking technology can be considered under two distinct headings:

1. Infrastructure networks
2. Ad Hoc networks

This breakdown into two distinct areas depends on the level of independence afforded to the mobile hosts in the network.

In an Infrastructure network, mobile hosts can communicate with mobile hosts inside or outside their local area by first communicating with a fixed base station. The base stations are connected to each other by some backbone network. Base stations route packets, which are destined for other mobile hosts, to the correct base station that then forwards the message to the destination mobile host. Well known examples of this type of network include existing GSM networks [17], 3G networks [10] and the Point Coordination Function (PCF) of 802.11 [12].

Each of these infrastructure networks have a number of problems with regard to the key properties mentioned in the introductory section. For example in terms of scale and time criticality, both GSM and 3G networks have limited bandwidth compared to existing wired networking technologies and large latency since they require

two mobile hosts in the same area to communicate through a fixed base station. Also in terms of scale, the PCF of 802.11 has been shown to perform badly [22]. Finally in terms of autonomy, a mobile host in an infrastructure network is required to be within communication range of a base station to be able to communicate with other mobile hosts.

What differentiates an ad hoc network from and infrastructure network is that there are no fixed base stations with which a mobile host communicates. Each host in the network is capable of moving. Obviously, as there are no base stations, mobile hosts are more independent and the onus is on the mobile host to discover other mobile hosts in the network and to cooperate with these mobile hosts to enable the delivery of messages in the network.

In terms of increasing scale and geographical dispersion, the workload on the mobile hosts in an ad hoc network increases correspondingly. In addition, mobile hosts in the ad hoc network need to reach consensus on the transmission of timely and safety critical messages to prevent the well known hidden terminal problem [21] where one mobile host corrupts the messages of another mobile host.

3.3 Related Work

In this section, we review some of the existing techniques used to control medium access in wireless ad hoc networks. Typically, there are two main approaches used to access the wireless medium in an ad hoc network:

1. Contention
2. Scheduled access

These two techniques will be covered with particular emphasis on their safety and timeliness properties.

3.3.1 Contention based approaches

In these approaches, mobile hosts contend with each other for access to the wireless medium. In MACA [14] sensing of the wireless medium before transmitting is not done. Each mobile host transmits a "Request To Send" (RTS) control packet causing mobile hosts that receive this packet to defer. The destination of the RTS packet replies with a "Clear To Send" (CTS) control packet. On receiving the CTS packet, the successful mobile host then sends its data packet.

There is a possibility of collisions of the RTS packets using MACA. If a mobile host does not receive the CTS packet correctly then it executes a binary exponential back-off algorithm.

MACAW [3] builds on MACA by introducing a Data-Sending (DS) packet to indicate to mobile hosts that the RTS/CTS packet exchange was successful. In addition, MACAW changes the back-off algorithm to include the current value of the back-off counter in the packet header field. Both 802.11 and FAMA-NTR [11] also build on top of MACA with each mobile host using carrier sensing before transmitting.

In each of these schemes, a mobile host does not have access to the wireless medium in a timely manner. In both MACA and MACAW, RTS control packets can

easily be corrupted due to two or more mobile hosts transmitting at approximately the same time. The possibility of this occurring in 802.11 is reduced (but not eliminated) by each mobile host waiting a random number of DIFS periods and sensing the wireless medium for other transmissions before transmitting.

In contrast to the above schemes, Sobrinho and Krishnakumar [20] use a black burst contention period to determine whether a mobile host gains access to the wireless medium or not. The mobile host sends a burst of energy of a known duration which is proportional to the amount of time that the mobile host has been waiting to access the medium. After transmitting the burst, the mobile host listens for other mobile hosts transmitting a longer burst. If there is a longer burst, then the mobile host defers its transmission. Otherwise the mobile host transmits its packet.

Markowski and Sethi [16] use a Contention Resolution algorithm (CRA) to support the co-existence of hard, soft and non real-time data. A mobile host transmits its packet (of a fixed size) and then listens for feedback information from other mobile hosts as to whether a collision has occurred or not. If a collision has occurred, then the CRA is executed. The CRA at each mobile host uses a window of size n (where n is the known number of hosts participating in the protocol) and divides this into an active part and an inactive part. A mobile host in the active window transmits while those in the inactive window defer. If another collision occurs, then the hosts in the active window are split again into an active and an inactive part. This continues until a mobile host can successfully transmit.

The main disadvantage of the Sobrinho and Krishnakumar and the Markowski and Sethi medium access schemes is that they assume that participating mobile hosts are all within communication range of each other. Therefore, neither of these protocols deal with the hidden terminal problem.

3.3.1.1 Scheduled based approach

In this approach, mobile hosts negotiate a set of slots for individual mobile hosts, to transmit in, so that these transmissions are as free of collisions as possible. Once a mobile host has a slot allocated to it, it then has access to the medium in a timely fashion. However, this may not prevent another mobile host from also transmitting in this slot and therefore corrupting the mobile hosts transmission. These schedule-based approaches can be broken into two categories: topology-dependent and topology transparent scheduling.

Topology-dependent scheduling assigns time slots to mobile hosts (or links between mobile hosts) within a two-hop neighborhood. As the topology changes, the time slots are reassigned in a distributed manner by the mobile hosts.

One approach, from Cidon and Sidi [8], uses a dedicated set of slots as a control segment to resolve conflicts and broadcast channel reservations. The number of dedicated slots used is proportional to the number of nodes in the network. Therefore as the number of mobile hosts in the network increases, the number of dedicated slots also increases.

Another topology-dependent approach, from Bao and Garcia-Luna-Aceves [15], uses identifiers for one-hop and two-hop neighbors to decide whether or not a mobile host (or a link between two mobile hosts) can use the current time slot. In addition to using one-hop and two-hop neighbor information, each mobile host uses a pseudo-random number generator (each mobile host using the same initial seed) to determine

which mobile hosts (or links) can be active in the current time slot.

From a timeliness point of view, whether or not a mobile host gains access to the time slot to transmit their packet depends on the current state of the pseudo-random number generator and the mobile hosts local two-hop topology information. Therefore a mobile host is unable to know when it will gain access to the wireless medium again. In addition, collisions can still occur using this approach when a mobile host does not have complete knowledge of the local one-hop and two-hop topology information, e.g. when network partitions begin to heal/merge.

In contrast to topology-dependent approaches, topology-transparent approaches, proposed by Chlamtac and Farago [7], and Ju and Li [13], allocate a collection of slots to a mobile host to use. The underlying idea of these approaches is that if a mobile host transmits in each of its allocated time slots then any neighbor of this mobile host will receive correctly at least one transmission from the mobile host. This is made possible by each mobile host using a unique code that determines which time slots a mobile host will use. However, one of the main limitations of the above two approaches is that the maximum number of neighbors of any mobile host is known and bounded. Another problem is that a transmitting mobile host does not know in which slot a particular neighbor can correctly receive their transmissions. Therefore, the mobile host must use all the slots allocated to it to ensure that at least one message is received correctly.

Another topology transparent protocol by Amouris [2] uses the position of a mobile host to determine in which slot that mobile host transmits in. In this work, space is divided into virtual geographic cells called space slots in a similar way to existing cellular networking techniques. A time slot is allocated to a space slot and due to spatial reuse, this time slot can be reused in space slots sufficiently far away. If there is more than one mobile host in a space slot, then the allocated time slot is used by the mobile hosts in a round-robin fashion by each mobile host maintaining a sorted list of the mobile hosts in the space slot.

As a mobile host moves from one space slot to another, it broadcasts a packet to inform mobile hosts in the new space slot and those in the old space slot of its arrival/departure. Another mobile host, in the new space slot, replies to this packet by transmitting a packet including the sorted list of mobile hosts located in the new space slot.

There are a number of problems with this protocol in terms of a mobile host having predictable access to the wireless medium. Firstly, the author of the paper does not specify how a mobile host that powers on in a space slot obtains a slot. A similar problem arises when two mobile hosts enter an empty cell or power on in an empty cell at the same time. These problems could easily be solved by allocating a number of time slots to allow mobile hosts to indicate their presence in the cell.

Another problem is that the sorted list of mobile hosts is replicated by each mobile host in the space slot (cell). Therefore, updates to this replicated data structure need to be carried out in a consistent manner, otherwise there is a possibility of the sorted list becoming inconsistent across mobile hosts eventually resulting in collisions in the assigned time slot. Finally, the total bandwidth is divided by the number of space slots in a virtual (space) frame. Thus if there are a large number of mobile hosts in a space slot and a smaller number of mobile hosts in each of the neighboring space slots then the larger number of mobile hosts are still limited to the one assigned slot.

3.4 MAC protocol for Ad Hoc networks

This section describes a new Medium Access Control (MAC) Protocol for Ad Hoc networks, called the TBMAC (for Time-Bounded Medium Access Control) protocol, which provides mobile hosts with predictable access to the wireless medium. This new protocol exploits geographical information to allow mobile hosts predictable access to the wireless medium.

3.4.1 Protocol Introduction

To provide each mobile host with predictable medium access, the TBMAC needs (i) to reduce as much as possible the possibility of the transmissions of two or more mobile hosts from colliding and (ii) to detect collisions when they occur and to take some action to prevent these collisions from recurring.

To reduce as much as possible the possibility of the transmissions colliding, the geographical area occupied by the mobile hosts is statically divided into a number of geographical cells. The cells can have arbitrary shape and size but for simplicity, let us assume that the cells are hexagons of equal size as illustrated in Figure 1. Each cell is numbered and each numbered cell is allocated a distinct CDMA spreading code (or radio frequency) to use. This allocation of codes to cells occurs statically and is done to allow channel re-use in a similar fashion to existing cellular networking techniques [18].

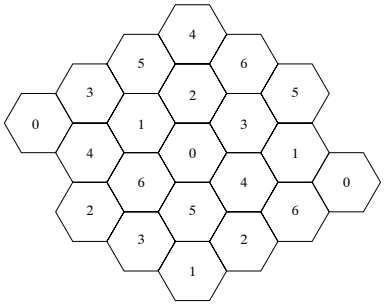


Figure 1: Possible Cell configuration

The motivation behind dividing the area of coverage into a collection of cells is to reduce the possibility of the hidden terminal problem [21]. In order to achieve this, the width of a cell is related to the transmission range of the wireless technology being used. By relating these, the probability of one mobile host hearing the transmission of another mobile host in the cell is increased thus reducing the possibility of the second mobile host in the cell beginning a transmission while the first mobile host's transmission is in progress.

The division into cells does not completely solve (i) and (ii) above. Collisions can still occur as two hosts could transmit simultaneously thus corrupting each other's packets at a receiving mobile host. When collisions do occur, we have not yet presented any mechanism to detect these collisions and to then prevent them from recurring (see section 3.4.1).

To further reduce the possibility of collisions, access to the medium within a cell is divided into two time periods:

1. Contention Free Period (CFP)

2. Contention Period (CP)

The division into a CFP and a CP is similar to the Point Coordination Function in the 802.11 standard [12] with the exception that the TBMAC CFP does not rely on one particular mobile host to act as an access point. Instead of using an access point, the TBMAC allows distributed agreement to be reached by the mobile hosts in a cell. Further details on how this agreement is achieved can be found in section 3.4.1

Both the CFP and the CP are divided into slots and each period lasts a well-known period of time. Once a mobile host has been allocated a slot in the CFP, it has predictable access to the wireless medium. The mobile host can then transmit data in its slot until it leaves the cell or fails. When a slot in the CFP is allocated to one mobile host in the cell, a mobile host sends a Null message in its slot even if it does not have a message to send.

Mobile hosts, that do not have CFP slots allocated to them, contend with each other to request CFP slots to be allocated to them in the CP. The CP is used by mobile hosts that have arrived into the cell or that have recently powered on in the cell. The steps required for a mobile host to be allocated a slot are covered in section 3.4.5.

Dividing access to the medium into two well-known time periods requires the clocks of all the mobile hosts in the network to be synchronised. Again equipping each mobile host with a GPS receiver would satisfy this clock synchronisation requirement. If GPS does not provide sufficient synchronisation precision, then a clock synchronisation protocol could be executed to synchronize the clocks of mobile hosts [19]. Periodically, in each cell at approximately the same time (based on the clock synchronisation precision), the CFP begins followed by the CP.

The rest of this section describes how the TBMAC protocol operates and is broken into the following parts:

1. Protocol Basics

2. Atomic Agreement

3. Communication Between Cells

4. Slot Allocation

5. Slot Deallocation

6. Protocol extensions

3.4.2 Protocol Basics

When a mobile host enters a cell and requires a slot in the CFP to be allocated to it, the mobile host first needs to learn whether there are other mobile hosts already in the cell with CFP slots allocated to them and what CFP slots have been allocated. For the moment, let's ignore the case where there are no mobile hosts in the cell with CFP slots allocated to them. This case will be covered in detail in section 3.4.5.

The newly arrived mobile host listens for one full CFP to pass before requesting a slot in the following CP. As part of each CFP slot message header (see Figure 2), there is an Allocated Slot Bitmap field. The Allocated Slot Bitmap field indicates the number of slots allocated in the cell. By receiving at least one message in the CFP correctly, the listening mobile host obtains the number of slots allocated and the position of these allocated slots in the CFP.

In addition to the Allocated Slot Bitmap field, a CFP message header also contains the Current Slot field which holds the value of the current slot being occupied and the type of the message being transmitted. In addition to these fields, the header contains an Extensions field (see section 3.4.7) and an Additional Info field. The Additional Info field is used for extra information depending on the type of the message being transmitted.

The Message Type field is used to indicate the type of message being transmitted. The different possible types of messages are:

1. Data
2. Acknowledgment
3. Null
4. Rebroadcast
5. Slot Allocation Request
6. Slot Deallocation Request
7. Inter-cell Communication Request

The need for the first two message types is relatively obvious. The Null message type was introduced in section 3.1. The remaining four message types are control messages.

How these control messages are used will be covered in the next sections. Briefly, the Rebroadcast message is used to achieve atomic agreement (see section 3.4.3), the Slot Allocation and Deallocation Request messages are used by mobile hosts to allocate and deallocate CFP slots (section 3.4.5 and 3.4.6). Finally, an Inter-cell Communication Request message is used when a mobile host wishes to communicate across its current cell boundary with a neighboring cell.

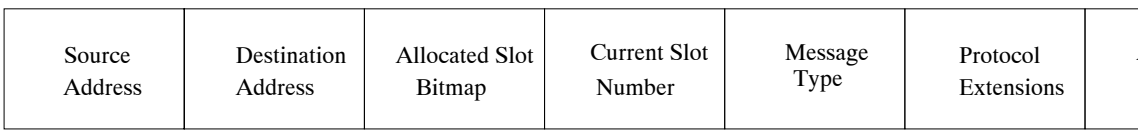


Figure 2: Frame Format Header

3.4.3 Atomic Agreement

To perform various actions within a cell, mobile hosts need to reach agreement with the other mobile hosts within the cell. In the PCF of 802.11, this agreement is

enforced by the Access Point that coordinates access to the wireless medium by polling mobile hosts for data to send. In the Ad Hoc environment that we are considering, we cannot assume there is an Access Point present in the area covered by the geographical cell.

One option would be to elect an Access Point from the mobile hosts present in the cell. This option has a number of problems. The mobile hosts have to reach a distributed agreement on which mobile host is to become the access point. Mobile hosts would also have to monitor the access point for failure and to reach agreement that the failure has occurred. This option has simply moved the problem of reaching distributed agreement to distributed leadership election and failure detection.

The second option would be to provide a total ordering protocol between the mobile hosts within a cell. Messages sent by a number of mobile hosts using a total ordering protocol are delivered by each mobile host in the same order [4]. Therefore, two messages, allocating slots for example, are seen in the same order by every mobile host in the cell. This is our preferred option as it distributes the task of slot allocation to the mobile hosts within a cell that have CFP slots allocated to them (note that we are ignoring the problem of reaching agreement between two or more mobile hosts that do not have slots allocated to them until section 3.4.5).

The approach we use to provide a total ordering protocol within a cell is to use the Synchronous Atomic Broadcast protocol from Flaviu Cristian [9]. Typical uses of the synchronous atomic broadcast protocol within a cell are to allocate and de-allocate slots in the CFP and for requests to communicate across cell boundaries.

To explain how the Synchronous Atomic Broadcast protocol works, consider a mobile host with a CFP slot that wishes another CFP slot to be allocated to it. Before the mobile host sends a Slot Allocation Request message using the Synchronous Atomic Broadcast protocol, it inserts a sequence number and a timestamp into the Additional info field of the message header. The mobile host then broadcasts the message a number of times using its existing CFP slot(s).

Since a mobile host with a CFP slot has predictable access to the medium, other mobile hosts in the cell will hear this transmission. When another mobile host with a CFP slot receives this message, if the mobile host has not processed the message before, based on the sequence number, then it stores the message and then rebroadcasts the message until the delivery time of the message arrives.

The delivery time of the message is equal to the original timestamp of the message plus the delay to delivery, Δ, which is a parameter of the Synchronous Atomic Broadcast protocol. For example, the Δ for the TBMAC protocol would typically be 2 * (CFPs + CPs). When the delivery time of the message arrives, all the mobile hosts in the cell then update their information consistently and allocate the mobile host a new slot.

The reason for sending a message a number of times (and other mobile hosts rebroadcasting the message) is to increase the probability of all the mobile hosts in the cell receiving this message.

On first reading of the above description, it would appear that if a mobile host wishes to reach agreement with the other mobile hosts in a cell then it needs to retransmit the same message a number of times in it's slot. However, since the information specific to a Synchronous Atomic Broadcast is relatively small, this information could easily be piggybacked on new data packets being transmitted.

3.4.4 Communication Between Cells

To allow inter-cell communication, a number of slots of the CFP are preallocated specifically for this task.

Figure 3: Slots for inter-cell communication

Since the area occupied by the mobile hosts has been divided up into geographical cells and each cell has been allocated a particular radio channel to use. There is an obvious problem of how a mobile host communicates with other mobile hosts in neighboring cells. A simple solution to this problem is to statically allocate two CFP slots of a particular cell for communication with mobile hosts in a neighboring cell.

One slot is used for communication across the cell boundary in one direction using the destination cells radio channel and the other slot is used for communication in the other direction. Referring to Figure 1 and Figure 3, the CFP of a cell would have 14 (or some multiple of 14 to allow more than one mobile host to communicate across cells during the CFP) of its slots pre-allocated for inter-cell communication.

When a mobile host wishes to communicate across cells, it atomically broadcasts a "Communication Between Cells" Request message to the other mobile hosts in the cell. Within this message, the sending mobile host includes the inter-cell slot (or slots) that it wishes to use. After the delivery of this message arrives, the mobile host sends its message in the inter-cell slot allocated to it.

3.4.5 Slot Allocation

There are two possible reasons why a mobile host in a cell may not have a CFP slot allocated to it. The mobile host could could have switched on after an idle period or the mobile host could be entering the cell from another cell. In each of these scenarios, there are two different possibilities. Firstly, there can be other mobile hosts already in the cell that have been previously allocated slots or there can be no mobile hosts in the cell. In total, we need to consider four possibilities:

1. Non-empty Cell / Mobile host powers on
2. Empty Cell / Mobile host powers on
3. Non-empty Cell / Mobile host enters cell
4. Empty Cell / Mobile host enters cell

3.4.5.1 Non-empty Cell / Mobile host powers on

Firstly consider a mobile host that powers on after an idle period in a cell with other mobile hosts that have slots in the CFP already allocated to them. When a mobile host powers on for the first time, it waits for one full CFP to pass before continuing. By listening to one full CFP, the mobile host can build up a picture of the mobile hosts already allocated slots in the cell. A mobile host only needs to receive one message correctly in the CFP to know the number of slots that have been allocated. The newly powered on mobile host then requests a slot to be allocated to it. It requests this slot by sending a message in the CP.

Recall from the Introduction that the CP, like the CFP, is divided into a number of slots of equal duration. A simple approach would be for a mobile host to request a slot by choosing a random slot in the CP and broadcasting to every mobile host requesting a slot to be allocated to it. Mobile hosts with slots allocated, that correctly receive this request, then atomically broadcast this request during the next 2 CFPs. After the delivery of this atomic broadcast message, the mobile host can then access its allocated slot in the following CFP.

With this approach, there is a possibility of two or more hosts powering on within a cell at approximately the same time and choosing the same slot in the CP in which to broadcast and thus corrupting each other's packets. Therefore, there is a non-deterministic aspect to the above approach. It is desirable from a predictability point of view to reduce this non-determinism as much as possible. One way to detect these collisions would be to increase the size of each slot in the CP to include a MAC level acknowledgment, similar to acknowledgments of unicast data packets in 802.11. Instead of broadcasting, a newly powered on mobile host would unicast its request in the CP to a mobile host that has already been allocated a slot. This latter mobile host would then immediately acknowledge the correct reception of this unicast before the end of the slot.

Another way to further reduce the possibility of collisions would be to introduce a random back-off mechanism before the recently powered on mobile host begins to transmit its request to be allocated a slot. This is similar to the collision avoidance mechanism in 802.11 with the only difference being that the back-off window is fixed in size.

A final way to reduce the possibility of collisions would be to further subdivide the cell as illustrated in Figure 4. Each subdivision of the cell would correspond to one or more slots in the CP. When a mobile host powers on, it calculates which cell and then which subdivision it is in. It then sends it's request in the slot in the CP corresponding to the subdivision that it is in. By increasing the number of subdivisions, we decrease the possibility of two or more mobile hosts powering on in the same subdivision and therefore reducing further the possibility of collisions in the CP. The obvious disadvantage with increasing the number of subdivisions is that the number of slots in the CP needs to be increased accordingly, therefore reducing the network throughput and increasing the time between CFPs.

3.4.5.2 Empty Cell / Mobile host powers on

By dividing access to the medium in the CFP and the CP, the most difficult task to solve is how to allocate the first mobile host in the cell a slot in the CFP. Once there

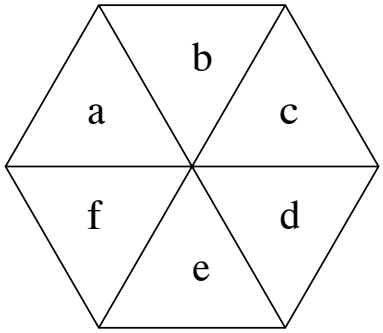


Figure 4: Subdivision of a Cell

is one mobile host in the cell with a slot in the CFP then it is possible to use the atomic broadcast protocol described in section 3.4.3 to allocate and de-allocate slots etc. A simple approach would be for a mobile host to wait for another mobile host to transmit in the CP. These two mobile hosts could then negotiate with each other in the CP to agree on an allocation of slots in the CFP. The main problem with this approach is that two mobile hosts could easily corrupt each other's packets in the CP and therefore be unable to reach agreement. The likelihood of this occurring increases as the number of mobile hosts powering on, within the cell, increases and is also exacerbated by the fact that the CP is relatively small compared to the CFP. The possibility of reducing these collisions could be achieved by using some of the techniques described in the previous section (see 3.4.5.1).

If there is the possibility of a large number of mobile hosts powering on at the same time, then the above approach could result in every mobile host in the cell being prevented from being allocated a slot in the CFP due to collisions during transmission in the CP. A different approach would be for a mobile host to choose a number of random slots in the CFP to transmit in. This approach is similar to the previously described approach with the advantage that the CFP is typically much larger than the CP and therefore the possibility of collisions is further reduced. By allowing collisions to occur in the CFP, we are contradicting our definition of the CFP as being free of contention. However, as we shall see in the following paragraphs, the possibility of these collisions continuing to occur after a brief number of CFPs is greatly reduced.

To clarify how the new approach would work, a mobile host powers on and listens for one full CFP (CFP 1 in Figure 6) in which the mobile host does not receive any transmission, the mobile host generates a list of slots to use during the next CFP. The success of this approach depends on the way the list of slots are generated. A simple algorithm for the generation of these slots would be to use a hashing function based on the MAC address of the mobile host to generate the list of slots to use. This hashing function could also take as a parameter the subdivision of the cell that the mobile host is in.

If two or more mobile hosts power on at the same time and in the same cell, then the probability of each mobile host generating the same list of slots to use in the next CFP is very small (by virtue of the hashing function). Thus during the next CFP (CFP 2 in Figure 6), one (or more) of the transmissions from one (or more) of the mobile hosts will be received correctly by the other mobile hosts.

This is illustrated in Figure 5 where three mobile hosts, A, B and C, generate a

list of slots to use. All but one of the slots generated by A, B and C are corrupted due to collisions. Therefore, A, B and C hear at least one transmission of the other mobile hosts.

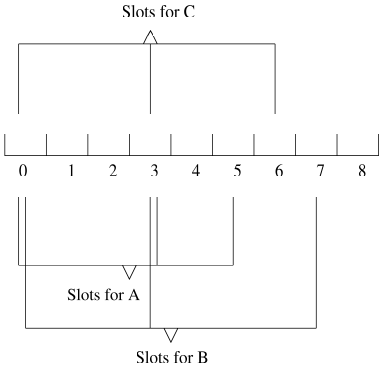


Figure 5: Collisions of generated slots

A naive option at this point would be for each mobile host that correctly receives a message from another mobile host to assume that the mobile host, which has successfully transmitted, has been allocated a slot and that the mobile hosts that have correctly received a message should then request a slot to be allocated to them in the next CP.

The reason that this option is naive is that may be possible for every mobile host correctly receives at least one message during the CFP (as in Figure 5). This would then result in every mobile host, that previously transmitted during the CFP, requesting a slot in the following CP because no mobile host knows if its transmissions have been successful or not and therefore no mobile host has been allocated a slot. In other words, every mobile host assumes that one of the other mobile hosts has been allocated a slot. This brings us back to where we were previously with no mobile host allocated a slot. Each mobile host would then have to generate another list of CFP slots and repeat the above sequence of steps.

The above deadlock problem can be solved by using the next CFP (CFP 3 in Figure 6) to transmit acknowledgments. During CFP 2, the mobile host transmits in the slots in its generated list and then listens for correct messages in each of the other slots. In the following CFP (CFP 3 in Figure 6), a mobile host transmits an Acknowledgment message, containing a Collision bitmap in the Additional Info field, in each slot in its generated list. The Collision bitmap represents CFP 2 with the position of messages, that were received by the mobile host correctly, being marked in the bitmap. By using the information in the Collision Bitmap and the Allocation Bitmap, each mobile host knows which of the messages in its various slots from CFP 2 were received correctly or were not received correctly (probably due to collisions). Therefore, at the beginning of CFP 4, each mobile host knows whether it can transmit successfully or not in each of its generated slots.

Each mobile host, that did not correctly receive an acknowledgment for any of its messages, requests a slot to be allocated to it in the next CP (CP 3 in Figure 6) according to the possible strategies described in section 3.4.5.1.

3.4.5.3 Non-empty Cell / Mobile host enters cell

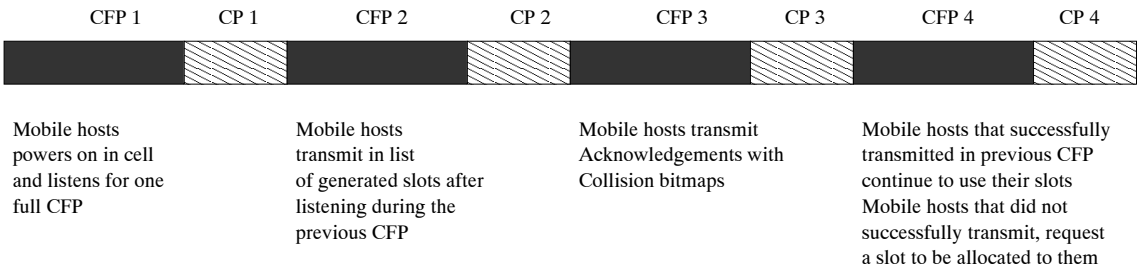


Figure 6: Using 4 CFPs to allocate slots in an empty Cell

A mobile host entering a cell, which has one or more mobile hosts allocated slots to them, is similar to mobile hosts powering on as described in section 3.4.5.1. The only difference is that the mobile host entering the cell has the possibility to communicate its impending arrival into the cell using one or more of the slots preallocated for inter-cell communication (see section 3.4.4). This allows mobile hosts, that are entering a cell from the same cell, to coordinate access during the CP. Thus, the inter-cell slot could be used by a mobile host to allocate a slot in the CFP of the cell that the mobile host is joining. When the mobile host enters the cell, it requests a slot in the CP but would get back an immediate acknowledgment from a mobile host that a slot has already been allocated to the arriving mobile host. Obviously, there is the possibility that the transmission of the arriving mobile host is not received correctly by any mobile host in the cell that the mobile host is joining. In this case, the arriving mobile host would revert to the steps outlined in section 3.4.5.1.

3.4.5.4 Empty Cell / Mobile host enters cell

A mobile host entering a cell, which has no CFP slots allocated to any mobile host, is similar to mobile hosts powering on as described in section 3.4.5.2. Similarly to section 3.4.5.3, an arriving mobile host could announce its presence using a preallocated inter-cell slot. This would allow other mobile hosts in the cell, both mobile hosts entering the cell and powering on, to avoid colliding with transmissions of this arriving mobile host when each mobile host begins the sequence of steps to allocate itself a CFP slot as described in section 3.4.5.2.

3.4.6 Slot Deallocation

In the previous section, we illustrated a variety of techniques for a mobile host to allocate itself a slot in the CFP. In this section, we describe how slots are deallocated, for example when a mobile host leaves the cell or fails.

The simplest way that a mobile host could deallocate one of its slots would be to atomically broadcast a message requesting the deallocation of its slot. This would typically happen after a mobile host has generated a list of slots to use as described in section 3.4.5.2 and does not need to use all the slots in the list. This could also happen if a mobile host is powering down or realises that it is leaving the cell based on location information and dead reckoning.

It is more difficult to deallocate a slot if the mobile host has failed or has left the cell without firstly deallocating it's slot. In addition, we would also like to avoid as much as possible deallocating a slot of a mobile host that does not want its slot

deallocated. Each mobile host, that has been allocated a slot, monitors each of the other mobile hosts slots in the CFP for correct reception of messages. If a mobile host does not correctly receive a number of messages from another mobile host, then it includes in the Additional Info field of each of its messages a bitmap indicating the positions of the messages that it has not received correctly. Each mobile host also checks the Additional Info field of each message from the other mobile hosts in the cell. After receiving a message from a majority of mobile hosts in the cell indicating that messages from another mobile host have not been received, then a mobile host atomically broadcasts a message requesting the slot (or slots) to be deallocated. After the delivery of this atomic broadcast message, the CFP slot(s) are deallocated by each mobile host in the cell.

3.4.7 Protocol Extensions

By fixing the number of slots in the CFP, it would appear that we are placing an upper bound on the number of mobile hosts that can be allocated a slot in the cell at any one time. This would mean that the TBMAC protocol is very restrictive and has only limited use.

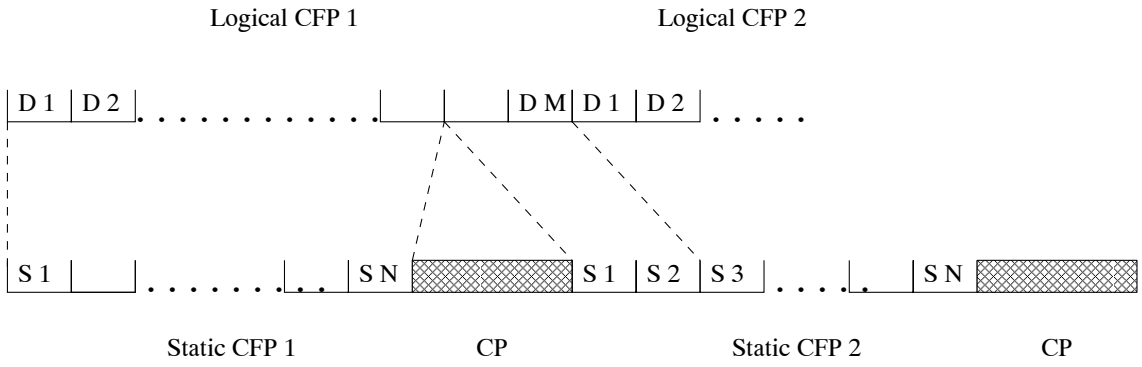


Figure 7: Mapping logical slots to static slots

However, it is possible however to overcome this restriction by allowing a dynamic logical CFP (LCFP) to grow and shrink in the repeating static CFPs. This is illustrated in Figure 7. Each static CFP has the same number of slots, N. The static CFP slots are numbered from S1 to SN.

By using the Atomic Broadcast protocol, the TBMAC can allow a LCFP to be mapped to a number of static CFPs. This is possible since each mobile host in the cell sees the same sequence of messages and therefore can allow the logical CFP to grow larger than N as more than N slots are allocated and to become smaller than N when slots are deallocated.

To allow the LCFP to grow and shrink, the TBMAC needs to include a Repeat Duration value in the Protocol extensions field in the header of each TBMAC message. The Repeat Duration field contains the number of slots before the allocation of slots repeats itself. Additionally, the size of the Allocated Slot Bitmap and the Collision Bitmap in the header of each TBMAC message would also need to grow and shrink.

As shown in figure 7, the LCFP is of size M and is mapped into two static CFPs (note that in this case N = M - 2). The dynamic slots, D1 to D(M-2) of the LCFP

are mapped into S1 to SN of the first static CFP shown. The last two dynamic slots, D(M-1) and DM, are then mapped to the following two slots of the next CFP, S1 and S2.

3.5 Acknowledgements

The work described in this paper was partly supported by the Irish Higher Education Authority's Programme for Research in Third Level Institutions cycle 0 (1998-2001) and by the Future and Emerging Technologies programme of the Commission of the European Union under research contract IST-2000-26031 (CORTEX - CO-operating Real-time senTient objects: architecture and EXperimental evaluation). The authors are grateful to past and current colleagues at Trinity College Dublin including Marc-Olivier Killijian as well as to Jörg Kaiser of the University of Ulm for their valuable input.

4 TCB Architecture and Protocols

4.1 Introduction

This section provides an overview of the Timely Computing Base model. Its properties and engineering principles are firstly described, followed by the basic services essential for timely and dependable computing. Lastly, we discuss the most relevant issues related to the application programming interface.

4.2 The Timely Computing Base Model

A system with a Timely Computing Base (TCB) is divided into two well-defined parts: a *payload* and a *control* part. The generic or *payload* part prefigures what is normally 'the system' in homogeneous architectures. It exists over a payload network and is where applications run and communicate. In particular, all middleware services dedicated to QoS provisioning, monitoring or management are constructed in the payload part of the system. The *control* part is made of local TCB modules, interconnected by some form of medium, the *control* network. Figure 8 illustrates the architecture of a system with a TCB.

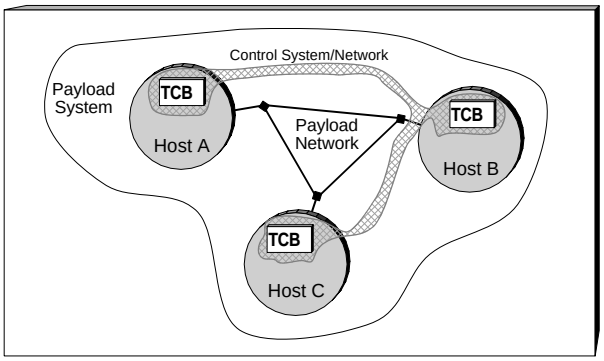


Figure 8: The TCB Architecture.

Concerning the payload part, the important property is that the system *can have any degree of synchronism*, that is, if bounds exist for processing or communication delays, their magnitude may be uncertain or not known. Local clocks may not exist or may not have a bounded rate of drift towards real time. The system is assumed to follow an omissive failure model, that is, components *only do timing failures*— and of course, omission and crash, since they are subsets of timing failures— no value failures occur.

In the control part, there is one local TCB at every node, fulfilling the following construction principles:

Interposition - the TCB position is such that no direct access to resources vital to timeliness can be made in default of the TCB

Shielding - the TCB construction is such that it itself is protected from faults affecting timeliness

Validation - the TCB functionality is such that it allows the implementation of verifiable mechanisms w.r.t. timeliness

TCB modules are assumed to be fail-silent, that is, they only fail by crashing. Moreover, it is assumed that the failure of a local TCB module implies the failure of that node. The TCB subsystem enjoys the following synchrony properties:

Ps 1 *There exists a known upper bound* $T_{D^1_{max}}$ *on processing delays*

Ps 2 *There exists a known upper bound* $T_{D^2_{max}}$ *on the drift rate of local TCB clocks*

Ps 3 *There exists a known upper bound* $T_{D^3_{max}}$ *on the delivery delay of messages exchanged between local TCBs*

Property **Ps 1** refers to the determinism in the execution time of code elements by the TCB. Property **Ps 2** refers to the existence of a local clock in each TCB whose individual drift is bounded. This allows measuring local durations, that is, the interval between two local events. These clocks are internal to the TCB. Property **Ps 3** completes the synchronism properties, referring to the determinism in the time to exchange messages among TCB modules. It is assumed that inter-TCB channels provide reliable delivery, that is, no messages addressed to correct TCBs are lost. The set of all local TCB modules, interconnected by the control channel, constitutes the distributed TCB. Note that the interposition, shielding and validation principles must also be satisfied by the distributed TCB.

Given the above set of construction principles and properties, a TCB can be turned into an oracle providing time-related services to applications or middleware components. To accomplish this, a set of minimal services has to be defined, as well as a payload-to-TCB interface.

4.3 TCB Services

In order to keep the TCB simple, the services defined are only those essential to satisfy a wide range of applications with timeliness requirements: ability to measure distributed durations with bounded accuracy; complete and accurate detection of timing failures; ability to execute well-defined functions in bounded time. Table 1 presents an informal summary of these services.

Service **TCB 1** allows the deterministic execution of some function given a feasible bound T, with the possibility of specifying an execution delay, as those resulting from timeouts. **TCB 2** allows the measurement of arbitrary durations with a known bounded error. If no external time sources are available, the measurement error will be proportional to the distance between the events, by a factor that depends on the drift rate of local TCB clocks ($T_{D^2_{max}}$). Finally, **TCB 3** and **TCB 4** describe the properties that a *Perfect Timing Failure Detector (pTFD)* should exhibit. We use an adaptation of the terminology of Chandra [6] for the timed versions of the completeness and accuracy properties. Although the timing failure detector service is constructed upon the other ones, it is essential for any useful TCB.

Timely Execution

TCB 1 Timely Execution: *Given any function f with an execution time bounded by T and a delay D, for any execution of f triggered at real time t the TCB will not execute f within D from t and is able to execute f within T from t.*

Duration Measurement

TCB 2 *Given any two events occurring in any two nodes at instants t_s and t_e, the TCB is able to measure the duration between those two events with a known bounded error. The error depends on the measurement method.*

Timing Failure Detection

TCB 3 Timed Strong Completeness: *Any timing failure is detected by the distributed TCB within a known interval from its occurrence.*

TCB 4 Timed Strong Accuracy: *Any timely action finishing no later than some know interval before its deadline is never wrongly detected as a timing failure.*

Table 1: Basic services of the TCB.

4.4 Programming Interface

Beside defining essential services to be provided by the TCB, it is very important to provide a programming interface to allow potentially asynchronous applications to dialogue with a synchronous component. From a practical point of view, the interface should be simple to allow an easy use of TCB services.

A relevant aspect to understand what can be done, is that applications or middleware components can only be as timely as allowed by the synchronism of the payload system. The TCB, although being a synchronous component, does not make applications timelier, it only provides the means to detect how timely they are. However, since it can detect timing failures, it may execute timely contingency plans, such as timely fail-safe shutdown, which is very relevant for the implementation of *fail-safe applications*. Another important aspect is that application components on top of the TCB are autonomous entities that take advantage of TCB services by construction. They typically use it as a pacemaker, letting it assess (explicitly or implicitly) the correctness of past steps before proceeding to the next step. This is relevant in the context of *time-elastic applications* since adaptation measures can be taken at these intermediate points, with the help of the TCB, to maintain required QoS coverage levels.

When defining an interface between an asynchronous and a synchronous environment, one of the most important problems is that the latency of service invocation, as well as the latency of service replies, may not be bounded. So it is not possible to relate (in a time line) events occurring in one side with events occurring in the other. The interface summarized in Table 2 makes a bridge between a synchronous environment and a potentially asynchronous one. Some examples of how to use this interface can be found in [5].

Duration Measurement

```
timestamp ← getTimestamp ()
id ← startMeasurement (start_ts)
end_ts,duration ← stopMeasurement (id)
```

Timely Execution

```
end_ts ← exec (start_ts, wait, exec_dur, f)
```

Timing Failure Detection

```
id ← startLocal (start_ts, spec, handler)
end_ts,duration,faulty ← endLocal(id)
id ← send (send_ts, spec, handler)
id,deliv_ts ← receive ()
id,dur_1,faulty_1 ··· dur_n,faulty_n ← waitInfo()
```

Table 2: Summary of the API.

The most basic function is `getTimestamp`, which allows an application to get a timestamp. With this single function an application is able to obtain an upper bound on the time it has needed to execute a computation step. It would suffice to request a timestamp before the execution and another after it. If this execution is a timed action, then the knowledge of this upper bound is also sufficient do detect a timing failure, should it occur. The `startMeasurement` and `stopMeasurement` functions are provided to do this in a more explicit way. A duration is measured by calling `startMeasurement` with a timestamp (previously obtained) that marks the start event. An `id` identifies the on-going measurement. The measurement terminates by issuing `stopMeasurement`, which returns the measured duration.

The timely execution of critical functions is provided through the `startExec` function. This is not a general purpose function. On the contrary, it is intended to be used only for sporadic actions with real-time requirements. When `startExec` is correctly used, the TCB will execute `func` accordingly to the parameters `start_ts`, `wait` and `exec_dur`. The first marks a reference point from where both the `wait` delay (deferral) and the maximum execution interval `exec_dur` should be counted. On return, a timestamp of the termination instant is provided through `end_ts`. It is obvious that not all `startExec` requests can be executed by the TCB, in which case an error status will be returned to the application. There must exist an admission control layer that performs the required admission tests before accepting any request. A more extensive discussion of this admission layer can be found in another paper [5].

The timing failure detection (TFD) service is presented to applications as a set of five functions. Two of them concern the detection of timing failures in local timed actions and the other three do the same for distributed timed actions. Note that failures in local actions are only important for the process performing them, while in the distributed case it is important to all those processes affected by the action.

To a certain extent, the `startLocal` and `endLocal` functions are similar to those of the duration measurement service. There are two new parameters: `spec` and `handler`. The former specifies the maximum execution duration and the later indicates an handler that is executed by the TCB, should a timing failure occur. As before, there is an `id` associated to each action under observation. With this

interface an application can be constructed to timely react to timing failures: in fact, it is the TCB who executes the handler as soon as a timing failure is detected. Note that even if the failure could be timely signaled to the application, there would be no guarantees about the timeliness of the reaction if it was done in the payload part of the system. When the timed execution finishes, the application has to call `endLocal` in order to disable detection for this action and to receive the measured duration (`duration`) and the timeliness status (`faulty`).

A distributed execution requires at least one message to be sent between two processes. Thus, in addition to local delays, the TFD service has to observe the delay of message delivery. This is done by intercepting message transmissions, in a very simple and intuitive manner, through the provision of `send` and `receive` functions. Note that for brevity reasons the function prototypes presented in Table 2 omit normal parameters such as addresses, message buffers, etc. The meaning of the `send` function parameters is identical to the ones of the `startLocal` function. We assume it is possible to multicast a message to a set of destination processes using this `send` function. A distributed duration is bounded by the `send_ts` and by a receive event generated within the TCB of a destination node. This means that each receiver will measure its own duration. All the observed durations for some message can be known by means of the `waitInfo` function. A process issuing this function will remain blocked until the TCB sends the information concerning some message. Although it would be possible to explicitly wait for the information concerning a specific message identified by `id` (as presented in [5]), this interface is more versatile and is therefore adopted.

Finally, note that the distributed duration measurement service is implicitly provided by the distributed TFD service.

References

[1] CORTEX Technical Annex Description of Work, October 2000.

[2] Konstantinos (Gus) Amouris. "Space-Time Division Multiple Access STDMA and Coordinated, Power-Aware MACA for Mobile Ad Hoc Networks". In *IEEE Symposium on Ad Hoc Wireless Networks (SAWN2001)*, San Antonio, Texas, 2001.

[3] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. "MACAW: a media access protocol for wireless LAN's,". *ACM SIGCOMM'94*, pages 212–225., 1994.

[4] Kenneth P. Birman. *"Building Secure and Reliable Network Applications"*. Manning Publications Co., 1996. ISBN 0137195842.

[5] A. Casimiro, P. Martins, and P. Veríssimo. How to Build a Timely Computing Base using Real-Time Linux. In *Proc. of the 2000 IEEE Workshop on Factory Communication Systems*, pages 127–134, Porto, Portugal, September 2000.

[6] Tushar Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, March 1996.

[7] Imrich Chlamtac and Andras Farago. "Making transmission schedules immune to topology changes in multi-hop packet radio networks". *IEEE/ACM Transactions on Networking*, 2(1):23–29, 1994.

[8] Israel Cidon and Moshe Sidi. "Distributed Assignment Algorithms for Multihop Packet Radio Networks". *IEEE Transactions on Computers*, 38(10):739–746, October 1989.

[9] Flaviu Cristian. "synchronous atomic broadcast for redundant broadcast channels". In *"Journal of Real-time Systems"*, pages 195–212. Kluwer Academic Publishers, 1990.

[10] ETSI. "Universal Mobile Telecommunications System (UMTS);3rd Generation mobile system Release Specification", 1999. www.3gpp.org.

[11] J. J. Garcia-Luna-Aceves. "Floor Acquisition Multiple Access (FAMA) for Packet-Radio Networks,". In *ACM SIGCOMM'95*, pages pp. 262–273., 1995.

[12] IEEE. "IEEE std 802.11 - wireless lan medium access control (MAC) and physical layer (PHY) specifications", 1997.

[13] J.H. Ju and V.O.K. Li. "An optimal topology-transparent scheduling method in multihop packet radio networks". *IEEE/ACM Transactions on Networking*, 6(3):298–306, June 1998.

[14] Phil Karn. "MACA - a new channel access method for packet radio.". In *ARRL/CRRL Amateur Radio 9th Computer Conference*, pages 134–140, Ontario, Canada, 1990.

[15] Bao Lichun and J. J. Garcia-Luna-Aceves. "A new approach to channel access scheduling for ad hoc networks". In *7th Annual International Conference on Mobile Computing and Networking*, Rome, Italy, 2001.

[16] Michael J. Markowski and Adarshapal S. Sethi. "Fully Distributed Wireless MAC Transmission of Real-Time Data". In *Fourth IEEE Real-Time Technology and Applications Symposium*, June 1998.

[17] M. Rahnema. "Overview of the GSM System and Protocol Architecture". *IEEE Communications Magazine*, 31(4):92–100, April 1993.

[18] Theodore S. Rappaport. *"Wireless Communications: Principle and Practice"*. Prentice Hall, 1996.

[19] Kay Romer. "Time Synchronisation in Ad Hoc Networks". In *The ACM Symposium on Mobile Ad Hoc Networking & Computing*, Long Beach, California, USA, October 2001.

[20] J. L. Sobrinho and A. S. Krishnakumar. "Real-time Traffic over the IEEE 802.11 Medium Access Control Layer". Technical report, Bell Labs, 1996.

[21] F. Tobagi and L. Kleinrock. "Packet Switching in Radio Channels: Part ii– The Hidden Terminal Problem in Carrier Sense Multiple Access and the Busy-Tone Solution". *IEEE Transactions on Communications*, 23(12):1417–1433, December 1975.

[22] M. A. Visser and M. El Zarki. "Voice and Data Transmission over an 802.11 Wireless Network". In *PIMRC'95*, pages 648–652, September 1995.