Coordination of Autonomous Mobile Entities

Mélanie Bouroche and Vinny Cahill
Distributed System Group, Computer Science Department, Trinity College, Dublin
{melanie.bouroche, vinny.cahill}@cs.tcd.ie

***Abstract*— Autonomous mobile entities, for example automated guided vehicles, are playing an increasingly important role in our everyday lives. Since these entities share their environment with each other and with humans, they need to coordinate their behaviour to ensure that strong safety constraints are respected. However, as sensing range and accuracy are inherently limited, and communication in wireless networks is unreliable, autonomous entities have access to only limited information about their environment and the current behaviour of other entities. This makes ensuring system-wide safety constraints particularly challenging.**

In this paper, we show how system-wide safety constraints can be translated into requirements on the behaviour of individual entities depending on the information available. We first present a formalism to express high-level system-wide safety constraints. We then introduce a notion of distributed responsibility allowing requirements on the behaviour of individual entities to be deduced. We show how this process can be applied to an example from the Intelligent Transportation System (ITS) domain.

I. Introduction

Progress in miniaturisation of computing devices is encouraging the deployment of autonomous mobile entities in our everyday environment. Examples of these entities include automated guided vehicles (AGVs) [2], and other mobile service robots [18], as well as robots for disaster rescue [6] and, in the future, autonomous cars. To ensure safe operation, such entities must coordinate their behaviour both with each other and with their environment. This means that the aggregated behaviour of all these entities must respect some system-wide safety constraints. As entities interact with their environment and are mobile, ensuring these system-wide safety constraints implies stringent real-time requirements on coordination. Furthermore, because the safety of humans and possibly crucial or expensive infrastructure is at stake, the coordination of entities is safety critical [9], i.e., a violation of the safety constraints could result in a catastrophe.

To coordinate their behaviour, entities can use both direct communication and communication via the environment. Because they are mobile, such entities will typically communicate over a wireless (possibly ad hoc) network. In wireless networks, however, communication, and in particular real-time communication, is highly unreliable and achievable performance varies greatly over time and location [5]. This implies that existing consensus-based coordination methods which rely on continuous real-time connectivity cannot be applied, and that entities need to take decisions independently. Ensuring that system-wide safety constraints will be ensured by entities taking decisions independently is particularly challenging. This is complicated by the fact that, as (i) the range and accuracy of sensors is inherently limited, (ii) communication in wireless networks is unreliable, and (iii) entities must act under stringent real-time constraints, entities must make decisions using only limited information.

Because the amount of information available to each entity varies over time, our approach is that entities should adapt their behaviour depending on the amount of information currently available to them. This includes information received by sensors and messages, but also information about the state of communication. The state of communication for an entity can be described as the identity of the entities with which it can communicate (as in group-based communication models [4], [10]), the achievable message latency (as in the quasi-synchronous communication model [19]), or the current proximity in which entities can communicate (as in the Space-Elastic Model [1]).

In this paper, we show how system-wide safety constraints can be translated into requirements on the behaviour of individual entities, by formalising safety constraints as sets of state incompatibilities that must be avoided and using a notion of distributed responsibility to attribute the responsibility for incompatibilities to individual entities. We then show how entities can prevent incompatibilities for which they are responsible, by adapting their behaviour depending on the amount of information available.

We first review existing work on the coordination of mobile autonomous entities and show that none of it explains how system-wide safety constraints can be ensured (Section II). We then present a formalism to specify system-wide safety constraints (Section III), and present the notion of distributed responsibility, and the coordination mechanisms that we have identified (Section IV). We eventually demonstrate how this process can be applied to an example from the Intelligent Transportation System (ITS) domain (Section V).

II. Related Work

The work of Nett et al. [13]–[15] aims to ensure reliable cooperation of autonomous vehicles, and, more generally, of autonomous mobile systems. Their approach is to provide an event service that delivers the global state of the system to all entities every time an event is raised. Entities can then use a local scheduling function to schedule shared resources. A number of coordination middlewares have also been proposed for mobile entities in ad hoc networks. LIME (Linda In Mobile Environment) [11], [12] is inspired by the Linda communication model, in which processes communicate through a shared tuple space. LIME caters for physical mobility of hosts and

logical mobility of agents, by having a tuple space attached to each mobile entity. Entities then collaborate by transiently sharing their tuple spaces, creating a "global virtual data structure". EgoSpaces [7], [8] is an extension of LIME, which defines the concept of a "view" that allows nodes to specify from which nodes tuples must be gathered. Finally, Limone (LIgthly-coordinated MObile NEtwork) [3] is another LIME-inspired middleware, designed for use on small devices and in unstable environments. All of this work focuses on providing a common view of the environment to all entities, but does not demonstrate that this approach can deliver the required level of reliability within the short time bounds available in the presence of unreliable communication. Furthermore, none of this work investigates how requirements on the behaviour of entities can be derived from system-wide safety constraints.

Other work [17] argues that there is a class of applications in which coordination needs to be ensured by the exchange of multiple, related, messages, i.e., a protocol. It is suggested that support for protocols should be offered at the middleware level to allow applications to use lower-level primitives such as identification and unicast. A middleware is proposed in [16], [17], based on the role abstraction, which specifies the behaviour of a class of interaction partners and allows support for a session. The authors do not describe how coordination guarantees could be met in the presence of unreliable communication. Furthermore, this work does not address how to derive requirements on entity behaviour, or how to design the coordination protocols.

III. Specifying Safety Constraints

In mobile settings, safety constraints typically include constraints on the states and actions of entities, as well as their proximity to each another. This exploits the rationale that entities need to coordinate their behaviour when they are in the same vicinity, the definition of which is application-specific. For example, two autonomous cars need to coordinate their behaviour only when they are close to each other, but not otherwise. In this section, we introduce a formalism to express these notions and their interactions.

A. Scenario, modes and states

A *scenario* encompasses a set of *entities* $E_1, E_2, .., E_n$, a *goal*, and some *safety constraints*. In the autonomous car example, the entities are the cars. The goal of this scenario is for the cars to drive to their destination. The safety constraint is that no car should collide with another car.

The behaviour of an entity is composed of a set of modes of operation (*modes*) that describe the actions it can take, and the transition rules between these modes. Modes should be defined so that an entity is always in one of its modes, i.e., transitions between modes are assumed to be instantaneous. For example, the modes of a car could be `going_at_Vmax`, `braking`, `stopped` and `accelerating`. We denote the set of modes of entity E_i as M_i.

The situation of an entity at a given time is described by its *state* which encompasses its mode, location, and some application-specific information about it. We denote the set of states of entity E_i as S_i. The state of a car is composed of its location, current speed, direction and mode. We define the function mode $\mathcal{M} : \cup \mathrm{S}_i \mapsto \cup \mathrm{M}_i$ that returns the mode of a given state.

B. States compatibility

We define a set of states $(s_1, s_2, ..., s_n) \in S_1 \times S_2 ... \times S_n$ as *compatible* if the safety constraints are not violated when some entities are simultaneously in these states. This relation will be noted $\mathcal{C}_\mathrm{S}(s_1, s_2, ..., s_n)$. For example, the states of two cars are compatible if they are far enough away, i.e., if the distance between their positions is bigger than a bound d.

C. Expressing the safety constraints

The safety constraints can be expressed as a set of incompatibilities between states, with constraints on the relative distance of entities (noted $distance(position1, position2)$). For example, the fact that two cars should not collide into each other could be expressed as:

$$s_{car1} \mathcal{C}_\mathrm{S} s_{car2} \\ \text{iff } \neg\big((distance(s_{car1}.position, s_{car2}.position) < d)\big) .$$

This example is simple, but illustrates that the formalism is high-level and implementation-independent. This formalism captures all the salient details of the safety constraints, and allows numerous safety requirements for mobile autonomous entities to be expressed simply. It also enables the geographical aspects of safety constraints to be captured.

IV. Translating safety constraints

High-level system-wide safety constraints, while being simple and quite intuitive to state, are not easily exploitable as such. In our experience, it is non-trivial to deduce the necessary and sufficient requirements on individual entity behaviour from such safety constraints, or even to check that some specification of the entity behaviour ensures that these safety constraints will not be violated. To ease this process, we introduce a number of concepts that can be used to derive requirements on entities.

A. Responsibility

For every possible incompatibility between the states of two entities, i.e., possible violation of one of the safety constraints involving these two entities, at least one of them needs to ensure that it will not occur. We say that this entity is *responsible* for the incompatibility.

The responsibility can be attributed to entities of a certain type or to entities in a certain role. For example, entities of the type traffic light might be responsible for ensuring that cars do not go through the crossing when the light is red. Another example might be that every car in the leading role (i.e., in front), be responsible for ensuring that no car collides into it from behind. Responsibility might be attributed a priori or in real-time, and might be transferred. However, at any time, at least one entity must be responsible for each possible

TABLE I
COORDINATION PRIMITIVES

Primitives	Meaning
Adapt	Perform another action than the one planned
Delay	Perform an action later than initially planned
Transfer responsibility	Communicate with other entities

incompatibility. When using the model, an initial partitioning of responsibility fulfilling this criteria must be chosen.

This notion of responsibility is the first step in the translation of system-wide safety constraints: it allows to distribute the duty of ensuring safety constraints over entities. Being responsible for an incompatibility implies requirements on the entity's behaviour: it should ensure at any time that the incompatibility will not happen. This requires that an entity be able to foresee when an incompatibility might happen. This can be deduced from the modes of the different entities. For this purpose, we define the notion of mode compatibility.

B. Mode compatibility

A set of modes $(m_1, m_2, ..., m_n) \in M_1 \times ... \times M_n$ is *compatible* if, when some entities are simultaneously in these modes, their states are compatible. If we define, for $m \in M_i$, $S_{i,m}$ as the set of states of the entity E_i in which it is in mode m, i.e., $S_{i,m} := \{s \in S_i | \mathcal{M}(s) = m\}$, mode compatibility can be defined as:

$$\mathcal{C}_{\mathrm{m}}(m_1, m_2, ..., m_n) \text{ iff}$$
$$\forall (s_1, s_2, ..., s_n) \in S_{1,m_1} \times ... \times S_{n,m_n}, \mathcal{C}_{\mathrm{s}}(s_1, s_2, ..., s_n)\,.$$

While the notion of state incompatibility captures whether the safety constraints are being violated at a given time, mode compatibility enables us to make predictions that no incompatibility will happen (while entities are in these given modes). It must be noted that if the modes of a set of entities are not compatible, it does not imply that the safety constraints will be violated. For example, the modes `stopped` of one car and `going_at_the_vmax` of another are not compatible, as entities might collide into each other when they are in these modes, but if they are far enough apart, the safety constraints will not be violated (and so their states at the time will be compatible).

C. Coordination primitives

For a responsible entity to ensure that no state incompatibility will happen, it is sufficient to ensure that its mode is at any time compatible with the modes of all surrounding entities. We have identified so far three different primitives that a responsible entity can use to this effect: delay its own action, adapt its behaviour, or communicate with other entities. Each of these mechanisms is detailed below and summarised in Table I. While other mechanisms might be found, our experience to date demonstrates that these allow a large number of scenarios to be solved.

1) Adapting its behaviour: A responsible entity can have information about the modes that other entities can be in both a priori (by previous knowledge) and in real-time, by means of message or sensor information. Using this information, a responsible entity can adapt its behaviour, i.e., perform an action other than the one planned, to always ensure it is in a mode in which the safety constraints will not be violated.

The *default modes* of an entity are the modes it can be in when it has no information about the current state of its environment. A mode m of an entity E is said to be a *fail-safe mode* if it is compatible with all the default modes of all the other entities. It is sufficient for an entity to remain in a fail-safe mode to ensure that the incompatibility that it is responsible for will not happen.

2) Delaying actions: The second primitive consists of delaying an action that can trigger an incompatibility (i.e., delay switching to a mode in which an incompatibility might occur). An entity can delay its action until it gets information that it is safe to undertake it, or until it has warned all entities that it will undertake it.

3) Transferring responsibility: Another means for responsible entities to ensure that the incompatibility for which they are responsible will not occur, is to warn other entities that the incompatibility might occur, by sending a message, which can include their state and mode. Entities that receive such a message become responsible to ensure that no incompatibility arises with the entity that sent it, which corresponds to a *transfer of responsibility*. The transfer only occurs if the message has been received, so an entity needs to know that it can communicate with another entity in order to transfer its responsibility. This transfer is only partial as the responsible entity remains responsible for the incompatibility in relation to other entities.

D. Contracts

A responsible entity can use a combination of the three coordination primitives mentioned above to ensure that the incompatibility for which it is responsible will not occur. This must be decided a priori, and can be seen as an implicit contract between the responsible entity and other entities. We have identified three types of contracts:

1) Contract without transfer: In this case, the responsible entity will not transfer its responsibility, and must always ensure, by adapting its behaviour if necessary, that the safety constraints are not violated. Other entities do not need to be aware of the contract, or even of the existence of the responsible entity.

For this contract, the only coordination primitives used are Adapt and Delay, and they are used only by the responsible entity.

2) Contract without feedback: In a contract without feedback, the responsible entity must warn other entities at least a predefined $t_{warning}$ duration in advance when the safety constraints are liable to be violated. Other entities must be able, at any time, to react (i.e., change their behaviour to

TABLE II

REQUIREMENTS IMPOSED BY THE THREE TYPES OF CONTRACTS

Type of contract	Requirements on the responsible entity	Requirements on other entities
without transfer	Adapt its behaviour or delay its actions to ensure that the incompatibility it is responsible for will not happen	None
without feedback	Communicate at least $t_{warning}$ in advance when the incompatibility is liable to occur	Be able at any time to adapt within $t_{warning}$ to a message from the responsible entity
with feedback	Communicate at least $t_{warning}$ in advance when the incompatibility is liable to occur Adapt to the feedback from another entity within $t_{warning} - t_{feedback}$	Be able at any time to adapt within $t_{warning}$ to a message from the responsible entity, or to communicate within $t_{feedback}$ to the responsible entity.

NB: In these contract, the times mentioned are times of delivery (as opposed to times of sending of messages).

ensure that no incompatibility will happen) within $t_{warning}$of a message from a responsible entity.

In this case, the responsible entity can use any of the three coordination primitives, while other entities can use only Adapt.

3) Contract with feedback: In this contract, the responsible entity must also warn other entities at least $t_{warning}$ in advance when the safety constraints are liable to be violated. In this case, however, entities can provide feedback to the responsible entity, when they cannot adapt their behaviour so that the safety constraints will not be violated. Therefore, the responsible entity must also be able to react, to ensure that no incompatibility will happen, to the feedback from another entity arising from its previous message, within $t_{warning} - t_{feedback}$. Other entities must be able at any time either to react within $t_{warning}$ to a message from a responsible entity, or to communicate within $t_{feedback}$ to this entity. This contract might include the exchange of further messages, but after the initial exchange the entities have discovered the presence of each other, and if necessary, the delay to exchange more messages can be included in the definition of $t_{warning}$.

In this case, both responsible and other entities can use any of the three mechanisms.

The requirements on both responsible and other entities for each of the types of contract are detailed in Table II. These contracts are decided a priori. The use of the three mechanisms by both responsible entities (R) and others (O) in the three contracts is described in Table III. It must be noted that safety constraint will be guaranteed only if the contract is respected. If it is believed that some entities might disregard safety constraints and not obey a contract, legislation might be introduced to enforce this. For example, it could be imposed that every autonomous car commercialised obey a number of contracts necessary for the safety of all road users.

TABLE III

USE OF THE PRIMITIVES BY THE CONTRACTS

Contract	Adapt	Delay	Transfer responsibility
without transfer	R	R	-
without feedback	R, O	R	R
with feedback	R, O	R, O	R, O

E. Zones

These contracts can be translated into geographical zones around entities: the safety zone and the consistency zone.

The states of all entities of a scenario must be compatible at all times. But the safety constraints actually impose constraints only on specific states, typically when two entities are "close" according to some application-specific definition. For this reason, we define the *Safety Zone* as the set of positions of entities where their states are liable to be incompatible with that of the responsible entity.

If a responsible entity foresees that an entity could be in a state that is not compatible with its own state when that entity enters its safety zone, the responsible entity can choose to transfer its responsibility, by sending a message. In this case, it must do so early enough, so that the incoming entity will have time to adapt its behaviour (either by not entering the safety zone, or by changing its mode) to prevent the incompatibility. The zone in which this must be achieved is called the *Consistency Zone* of the mode m that the responsible entity is in, and noted $CZ(m)$. Its size can be expressed as:

$$CZ(m) = SZ + O\_reaction(m) \cdot v_{max}(m) \qquad (1)$$

where m is the mode of the responsible entity, $O\_reaction(m)$ is the maximum time necessary for an entity receiving a message indicating that another entity is in a mode m to ensure that its state will be compatible with that of the entity, and $v_{max}(m)$ the maximal speed at which entities might approach an entity which is in mode m.

F. Summary

In Section III, we have shown how system-wide safety constraints for mobile autonomous entities can be specified. We have then shown how system-wide constraints can be distributed among entities, using a notion of responsibility. Requirements on the behaviour of entities can be met using three identified coordination mechanisms, combined in contracts between entities. These contracts can be translated into geographical zones, hence scaling down the coordination problem, and the requirement on communication. The next section demonstrates how the coordination model can be applied on a specific example.

It must be noted that given a set of safety constraints, and some characteristics of entities, not every scenario is solvable when trivial non-progress making solutions (e.g., all entities idle) are not considered. The resolvability of a scenario can be assessed, but this is outside the scope of this paper.

V. Example: Emergency Vehicle Arrival Warning

In this section, we show how the coordination model can be applied to design a system to warn cars when an emergency vehicle is approaching, so that they can get out of its way, if possible.

A. Specifying the safety constraint

This scenario contains two types of entities: cars and emergency vehicles. In the following, we consider only interactions between emergency vehicles and cars (and ignore interactions between cars and between emergency vehicles). The goal of the scenario is for emergency vehicles to travel as fast as possible, under the safety constraint that they should not crash into cars. Communication can be used in addition to sensor data, to improve achievable speeds. Emergency vehicles and cars can send messages to each other, either in ad hoc mode, or using wireless infrastructure. Communication, however, is not always available, and its performance may vary greatly over time and location.

Given the maximum speed of an emergency vehicle v_{max}, and an increasing set of speeds $\{v_i\}_{i\in[1,n]}$ with $v_n = v_{max}$, the modes of emergency vehicles can be defined as: `stopped`, {`going_at_Vi`, `accelerating_to_Vi`, `braking_to_Vi`}$_{i\in[1,n]}$. The state of an emergency vehicle encompasses its mode, position, and speed.

The behaviour of a car can be modelled with the modes `travelling`, `getting_out_of_the_way`, and `out_of_the_way`. Its state encompasses its mode, position, and speed. The default mode of a car is `travelling`. The mode `stopped` of an emergency vehicle is compatible with the mode `travelling` of a car (assuming that cars can avoid crashing into a stopped emergency vehicle, as into any other obstacle), therefore `stopped` is a fail-safe mode for emergency vehicles. This captures the fact that it is sufficient for an emergency vehicle to be stopped to ensure that it will not collide into a car.

Using these definitions, the safety constraint that cars and emergency vehicles should not collide can be stated as:

$$\mathrm{s}_{car}\mathcal{C}_{\mathrm{s}}\mathrm{s}_{ev} \text{ iff } \neg\Big(\big(distance(\mathrm{s}_{car}.\text{position}, \mathrm{s}_{ev}.\text{position}) < d\big)\wedge (\mathrm{s}_{ev}.\text{mode} \neq \texttt{stopped})\wedge(\mathrm{s}_{car}.\text{mode} \neq \texttt{out\_of\_the\_way})\Big)$$

where the index "ev" refers to the emergency vehicle. This captures that the states of a car and an emergency vehicle will remain compatible, unless they are closer that a distance d apart, that the emergency vehicle is not stopped, and that the car is on the road (we consider only single-lane roads for explanation purposes).

B. Requirements on entity behaviour

In this part, we derive the requirements on the behaviour of cars and emergency vehicles so that the safety constraint can be ensured. In this scenario, we chose that emergency vehicles are responsible for preventing incompatibilities with cars. This choice is motivated by the fact that emergency vehicles can adapt their behaviour (by varying their speed) to the current state of communication. They need to transfer their responsibility to cars with which incompatibilities might happen or to remain stopped, so that no incompatibility will happen. This implies that an emergency vehicle should send a message to all the cars that are in front of it, so that they can get out of its way. It must ensure that it will warn the cars early enough so that they will have time to get out of its way before it arrives. On the other hand, cars must ensure that they will, at any time, be able to get out of the way of the emergency vehicle within the given time delay, or be able to send a message back to the emergency vehicle, to warn it that they cannot. Emergency vehicles and cars therefore obey a contract with feedback, with $t_{warning} = max(O\_reaction, feedback\_time)$ and $t_{feedback} = braking\_time$, where $O\_reaction$ is the time within which a car should get out of the path of an emergency vehicle, $feedback\_time$ the time required for a car to send feedback to the emergency vehicle, and $braking\_time$ the time required for an emergency vehicle to stop.

The safety zone SZ of an emergency vehicle can be defined as a circle of 2 m of radius around it. From (1), it can be derived that an emergency vehicle in the mode `going_at_Vi` needs to send messages in a zone

$$CZ(v_i) = SZ + O\_Reaction \cdot v_i\,.$$

Therefore, emergency vehicles need to adapt their speed so that they always travel at a speed v_i, such that they can warn entities within $CZ(v_i)$ $t_{warning}$ before they arrive. So at any time an emergency vehicle needs to adapt its speed to the current state of communication and to possible feedback from cars.

C. Evaluation

We have shown that a high-level system-wide safety constraint such as "emergency vehicles should not collide into cars" can be translated into requirements on the behaviour of autonomous entities. Provided that both emergency vehicles and cars respect the contract outlined above, they will never crash into each other, even if the quality of communication varies.

This solution maps the current operating protocol of emergency vehicles, where a siren warns cars that an emergency vehicle will arrive, and the emergency vehicle driver accelerates as much as possible provided that he can see that the road is free far enough to allow it to slow down if necessary. The proposed solution, however, allows the warning to be forwarded by the infrastructure or other cars and enables cars to warn the emergency vehicle if they will not be able to get out of its way. The range of communication can be far greater than the range of sight, so the addition of the proposed solution might allow an emergency vehicle to travel faster.

VI. Conclusion

In this paper, we have presented a coordination model for autonomous mobile entities. A formalism to express system-wide safety constraints for applications composed of autonomous mobile entities was first presented. We have then shown

how these constraints can be translated into requirements on the behaviours of individual entities. This approach has been demonstrated in an example.

This coordination model was designed specifically to take into account that the amount of information available to autonomous mobile entities is limited and varies over time. It allows entities to adapt their behaviour depending on the information available, including the state of communication. We have successfully applied this model to a variety of scenarios, including both mobile and stationary components. Our future work includes deriving lower-level and more detailed requirements on the behaviour of entities from the safety constraints for different communication models.

ACKNOWLEDGEMENT

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

REFERENCES

[1] Mélanie Bouroche, Barbara Hughes, and Vinny Cahill. Building reliable mobile applications with space-elastic adaptation. In *Mobile Distributed Computing workshop (MDC 2006)*, June 2006. to appear.

[2] Jack Cawkwell. A visually guided agv for use as passenger transport in urban areas. In *Proceedings of 2000 Intelligent Transportation Systems Conference*, pages 311–315. IEEE Computer Society, October 2000.

[3] Chien-Liang Fok, Gruia-Catalin Roman, and Gregory Hackmann. A lightweight coordination middleware for mobile computing. In Rocco De Nicola, Gian Luigi Ferrari, and Greg Meredith, editors, *Proceedings of the 6th International Conference on Coordination Models and Languages (COORDINATION 2004)*, volume 2949 of *Lecture Notes in Computer Science*, pages 135–151. Springer, February 2004.

[4] Roy Friedman. Fuzzy group membership. In André Schiper, Alexander A. Shvartsman, Hakim Weatherspoon, and Ben Y. Zhao, editors, *Future Directions in Distributed Computing*, volume 2584 of *Lecture Notes in Computer Science*, pages 114–118. Springer, June 2003.

[5] Gregor Gaertner and Vinny Cahill. Understanding link quality in 802.11 mobile ad hoc networks. *IEEE Internet Computing*, 8(1):55–60, January–February 2004.

[6] Shigeo Hirose and Edwardo F. Fukushima. Development of mobile robots for rescue operations. *Advanced Robotics*, 16(6):509–512, September 2002.

[7] Christine Julien and Gruia-Catalin Roman. Egocentric context-aware programming in ad hoc mobile environments. In *Proceedings of the 10th ACM SIGSOFT symposium on Foundations of software engineering (SIGSOFT '02/FSE-10)*, pages 21–30. ACM Press, November 2002.

[8] Christine Julien and Gruia-Catalin Roman. Active coordination in ad hoc networks. In Rocco De Nicola, Gian Luigi Ferrari, and Greg Meredith, editors, *Proceedings of the 6th International Conference on Coordination Models and Languages (COORDINATION 2004)*, volume 2949 of *Lecture Notes in Computer Science*, pages 199–215. Springer, February 2004.

[9] Hermann Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publisher, 1997.

[10] Jun Luo, Patrick Th. Eugster, and Jean-Pierre Hubaux. Pilot: Probabilistic lightweight group communication system for ad hoc networks. *IEEE Transactions on Mobile Computing*, 3(2):164–179, April 2004.

[11] Amy L. Murphy and Gian Pietro Picco. Using coordination middleware for location-aware computing: A lime case study. In Rocco De Nicola, Gian Luigi Ferrari, and Greg Meredith, editors, *Proceedings of the 6th International Conference on Coordination Models and Languages (COORDINATION 2004)*, volume 2949 of *Lecture Notes in Computer Science*, pages 263–278. Springer, February 2004.

[12] Amy L. Murphy, Gian Pietro Picco, and Gruia-Catalin Roman. Lime: A middleware for physical and logical mobility. In *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS'01)*, pages 524–533. IEEE Computer Society, April 2001.

[13] Edgar Nett, Martin Gergeleit, and Michael Mock. Mechanisms for a reliable cooperation of vehicles. In *Proceedings of the 6th IEEE International Symposium on High-Assurance Systems Engineering (HASE'01)*, pages 75–81. IEEE Computer Society, October 2001.

[14] Edgar Nett and Stefan Schemmer. Reliable real-time communication in cooperative mobile applications. *IEEE Transactions on Computers*, 52(2):166–180, February 2003.

[15] Edgar Nett and Stefan Schemmer. An architecture to support cooperating mobile embedded systems. In *Proceedings of the 1st conference on Computing Frontiers (CF'04)*, pages 40–50. ACM Press, April 2004.

[16] Kurt Schelfthout and Tom Holvoet. Coordination middleware for decentralized applications in dynamic networks. In *Proceedings of the 2nd international doctoral symposium on Middleware (DSM'05)*, pages 1–5. ACM Press, November 2005.

[17] Kurt Schelfthout, Danny Weyns, and Tom Holvoet. Middleware for protocol-based coordination in dynamic networks. In *Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing (MPAC '05)*, pages 1–8. ACM Press, November 2005.

[18] Rolf Dieter Schraft. Mechatronics and robotics for service applications. *IEEE Robotics & Automation Magazine*, 1(4):31–35, December 1994.

[19] Paulo Verissimo and Carlos Almeida. The timely computing base model and architecture. *IEEE Transactions on Computers - Special Issue on Asynchronous Real-Time Systems*, 51(8):916–930, August 2002.