Building Reliable Mobile Applications with Space-Elastic Adaptation

Mélanie Bouroche, Barbara Hughes and Vinny Cahill
Distributed System Group, Computer Science Department, Trinity College, Dublin
{melanie.bouroche, barbara.hughes, vinny.cahill}@cs.tcd.ie [*]

Abstract

Mobile applications, for example mobile robots, are playing an increasingly important role in our everyday lives. Since components of these applications share their environment with each other and with humans, they need to coordinate their behaviour to respect strong safety constraints. Unfortunately, they typically make use of wireless networks in which real-time communication is highly unreliable, making coordination particularly challenging.

We present a real-time communication model for wireless networks that takes into account that communication might not be reliable. It provides feedback to mobile components about the state of communication, so that they can adapt their behaviour accordingly. We show how this model can be used to build reliable mobile applications in wireless networks: this involves specifying the safety constraints that need to be respected, and translating them into constraints on the behaviour of individual components. To illustrate our approach, we describe an example from the Intelligent Transportation Systems domain.

1. Introduction

With the emergence of the ubiquitous computing vision, we will rely more and more heavily on computer-enabled devices in our everyday environment. Such applications will be composed of mobile, autonomous components, which are increasingly able to do complex tasks and act on their environment. Examples of such applications are automated guided vehicles (AGV) [1], and other service robots [13], as well as robots for disaster rescue [5] and, in the future, autonomous cars. Components of these applications share their environment with other components and humans; therefore, they need to coordinate their behaviour with each other and their environment, to ensure the safety of all the parties involved. Because the safety of humans and possibly crucial or expensive infrastructure is at stake, a delay in coordination can result in a catastrophe. Thus, this coordination exhibits hard real-time requirements [7].

To coordinate their actions and ensure that safety constraints are respected, components can send messages to each other. Autonomous mobile components however typically communicate over wireless networks, where communication is inherently less reliable than in wired networks because of the higher rate of link failures due to node mobility and varying signal strength. A number of attempts have been made to adapt the techniques used in more reliable networks to wireless ones. However, these techniques are designed assuming continuous connectivity and typically provide guarantees only when some assumptions about communication hold. For example, a commonly required assumption (e.g., in [11, 12]) is that communication is sufficiently good, in other words that there exists a so-called omission degree, that denotes an upper bound on the number of losses that may affect a single message. This assumption, however, might not always be fulfilled in wireless networks, hence the reliability needed for such applications might not be provided. Our approach is to take into account that communication might not be reliable, and provide feedback to application components about the state of communication, so that they can adapt their behaviour accordingly.

In this paper, we first present a real-time communication model for wireless networks, in which every communicating application component - or entity - is informed about the state of communication, in terms of the geographical area around them in which real-time communication can be guaranteed. In this model, messages are addressed to (interested) entities within an area, as opposed to a priori identified entities. The model is termed *space-elastic*, as the space in which real-time communication is guaranteed (within an application-specified delay) varies over time. Us-

[*]This paper was presented at the Mobile Distributed Computing workshop (MDC 2006) ©IEEE

ing this feedback, entities can adapt their behaviour as a function of the state of communication, in real-time. In the second part of the paper, we demonstrate how the model can be used in an example, first showing how safety constraints can be specified, and then how they can be translated into constraints on entity behaviour. We conclude the paper with a discussion of related work.

2. Space-Elastic Model

To allow application components to make progress in the presence of unreliable communication, we have designed a communication model in which feedback about the state of communication is provided to message senders. This model exploits the rational observed in [4], i.e., the relevance of context to a particular geographical area, to guarantee real-time communication within a geographical proximity only. This proximity can be defined either absolutely (via GPS coordinates), or relatively around the entity (using an anchor point and a size). In this model, messages are sent to (interested entities in) a geographical area as opposed to a set of a priori identified entities. The area in which timely communication is guaranteed will vary over time depending on the dynamics of the network. Entities sending a message are notified within a given delay of the area in which this message has been delivered. They can then adapt their behaviour depending on where the message has been delivered.

2.1. Specifications

An entity wishing to send a message specifies the geographical area in which it wishes this message to be delivered. This area is called the *Desired Coverage* (DC), and is used to bound message propagation. The sender also specifies the maximum latency *msgLatency* within which the messages must be delivered.

Depending on the state of communication (presence, density and partitioning of nodes, and quality of the wireless links), it might not be possible during some period of time to deliver a message to all interested entities within the desired coverage. Therefore, the size of the area in which timely delivery of messages is provided, called the *Actual Coverage* (AC(t)), will change over time. In the worst case, no communication is possible; this corresponds to AC(t)=0. The sender is notified of changes in the size of the actual coverage, within a bounded time, $adaptNotif$. Therefore, an entity knows within $msgLatency + adaptNotif$ after sending a message, the area in which it has been delivered.

Depending on the feedback about the state of

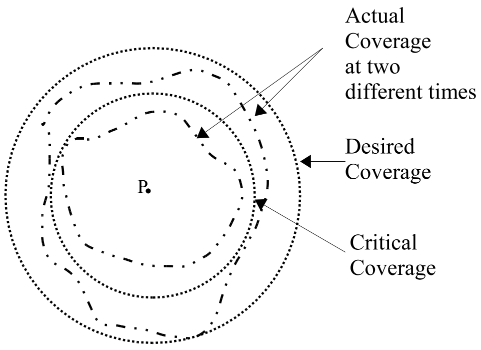

Figure 1. Different coverages of the Space Elastic Model

communication, the sender can adapt its behaviour. If the actual coverage becomes smaller than one or more thresholds (called *Critical Coverage(s)* (CC)), the sender might need to take into account that it cannot communicate in a area wide-enough to maintain safe operation, and might need to adapt its behaviour. Variations of the actual coverage around the desired and the critical coverage are shown in Figure 1.

2.2. Guarantees

In this model, guarantees about real-time communication are provided to both message sender and receivers. Senders are guaranteed to be able to communicate with a given latency within a specified coverage, and to be notified if this coverage changes, within a given time delay. On the other hand, entities present within the actual coverage at the delivery time of a message of a type in which they have expressed interest, are guaranteed to receive it. We define that an entity is present within the actual coverage once it is able to receive messages after arriving in the communication coverage. This will take an implementation-dependent time, *present*, which might be necessary to include the entity in the real-time route for example. We will see in the next section that these guarantees are easily exploitable to ensure system-wide safety constraints while allowing progress of entities.

2.3. Implementation

An implementation of the Space-Elastic Model, in the form of an event-based middleware called RT-STEAM [6, 10], has been designed and is currently being implemented. RT-STEAM is a real-time version of STEAM (Scalable Timed Events And Mobility) [9], which uses the SEAR (Space-Elastic Adaptive Rout-

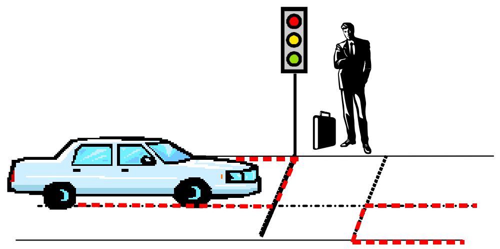

Figure 2. Pedestrian traffic light

ing) real-time routing and resource reservation protocol, over the TBMAC (Time-Bounded Mac Access Control) protocol [2]. While implementation details are not within the scope of this paper, it might be noted that the implementation is advanced enough to demonstrate that this model is implementable.

3. Specifying the Safety Constraints

In the remainder of this paper, we show how to use the space-elastic model to build reliable safety-critical applications in wireless networks. We demonstrate the process using a scenario from the traffic management domain.

In the automotive industry, much work has been done on intelligent, computer-assisted driving. The goals of these research efforts include enhanced comfort of both driver and passengers, and more importantly, improved safety and a better utilisation of road resources to alleviate congestion. The next step in this direction is the development of autonomous cars. We can imagine a pedestrian traffic light that would warn cars when pedestrians need to cross the street. This solution might be more efficient than using sensors only, and would be applicable on protected road stretches.

3.1. Scenario Description

We assume that cars and traffic lights are fitted with GPS and wireless communication facilities. It is not assumed that cars are aware of the position of traffic lights a priori. The protocol for safe driving (following the road and avoiding collisions with other cars) is outside the scope of this scenario.

The goal of this scenario is for the traffic light to turn red - to allow pedestrians to cross - as soon as possible after they press the request-to-cross button. In this scenario, the safety constraint is that no pedestrian be killed when crossing at a red light. This requires that no car passes through a red traffic light. We can also define some liveness requirements such as the fact that pedestrians should be able to cross as soon as possible, and cars should make progress. This enables us to exclude the simplistic solution that cars remain stopped. This example includes both fixed and mobile entities in a safety critical application, and is therefore representative of a class of application based on mobile autonomous entities.

3.2. Actions and States

A first step in building an application is to enumerate the different types of entities taking part in the application, as well as their possible states and actions.

This scenario involves two types of entities: traffic lights and cars. Without loss of generality, we consider only the states red and green for the traffic light (in this description, when we mention the colour of the light, it is always the one intended for cars and not pedestrians). Periodically, the traffic light reassess it state; it can choose to either stay in its current state or to switch to the other state. Therefore their are four possible actions for the traffic light: *(i)* switching from red to green, *(ii)* staying green, *(iii)* switching from green to red, and *(iv)* staying red. We assume that cars, on the other hand, can either *(i)* travel at its maximum speed, *(ii)* brake, *(iii)* be stopped or *(iv)* accelerate. The state of a car can be described as the action that it is undertaking, its position, speed and direction.

3.3. Compatibility

We will say that two states or actions are compatible when entities in these states (or undertaking these actions) do not violate the safety constraints. The safety constraints can then be specified using the compatibility of states of entities.

In our example, the safety constraint is that no car should go through a red light. So this means that if the traffic light is green, its state is compatible with all the possible states that a car can be in. Similarly, if a car is stopped, it state is compatible with all possible states that a traffic light can be in. Such actions are called fail-safe.

On the other hand, if the traffic light is red, (i.e., if it is either remaining red or switching to green), the safety constraints might be violated, if a car is too close to the traffic light and is not stopped. Therefore the actions remaining red for the traffic light and going at its maximum speed for the car, for example, are not compatible. The compatibility of the different actions of traffic light and car entities are summarised in Table 1.

The states of a car going at its maximum speed and of a red traffic light can be compatible, however, if they

Table 1. Action compatibility matrix for the traffic light scenario

	Remaining green	Switching to red	Remaining red	Switching to green
Going at Vmax	✓	✓	χ	χ
Braking	✓	✓	χ	χ
Stopped	✓	✓	✓	✓
Accelerating	✓	✓	χ	χ

are far enough apart. In this case, the car is not concerned by the traffic light. This is an example of the relevance of proximity for entities: entities are only interested in the actions of entities around them. The states of the car and the red traffic might become incompatible if they come closer, this is why their actions are not compatible.

Using the notion of states and action compatibility, we can specify the safety constraints of this scenario, in terms of incompatibilities that should be avoided: the states of a car and a traffic light are compatible unless they are close (for some application-specific definition), the traffic light is red, and the car is not stopped.

4. Translating the Safety Constraints

We have seen in the previous section that the safety constraints can be specified in terms of action and state incompatibilities that it is sufficient to avoid to ensure that the safety constraints will not be violated. We present in this section how this can be achieved, through a notion of responsibility.

4.1. Defining Responsibility

When an incompatibility between the states of two entities can arise, at least one of these entities should ensure that it does not. We say that this entity is responsible for this incompatibility. A responsible entity should be assigned for every possible inconsistency. Responsibility can be attributed to entities of a certain type, or to entities in a certain role. This notion allows a distributed enforcement of the safety constraints, as the system-wide safety constraints are translated into requirements on individual entities behaviours, without the need for a central component.

To ensure that the incompatibility for which they are responsible will not occur, responsible entities have three possibilities: they can delay their actions, adapt their behaviour, or transfer their responsibility. We will detail each of these possibilities below.

4.1.1. Delaying actions. A mechanism for a responsible entity to ensure that the incompatibility it is responsible for will not happen, is to delay an action that can trigger this incompatibility. It can delay its action until it gets information that it is safe to undertake it, or until it has warned all other entities that it will undertake it.

4.1.2. Adapting its behaviour. A responsible entity can adapt its behaviour, i.e., perform an action other than the one planned, to ensure that at all times the incompatibility for which it is responsible will not occur. For this purpose, it can use information about its environment and other entities (information known a priori, received in messages, or acquired by sensors).

4.1.3. Transferring its responsibility. An entity can ensure that no incompatibility will happen by sending messages to other entities, informing them about a possible incompatibility. Using the space-elastic model, the entity will send the messages to all interested entities in its vicinity. This is particularly appropriate, as entities that should be informed of the possible incompatibility might not be identified a priori.

After sending a message, the entity is notified in real-time about the area in which it has been delivered, and can assess whether this area is sufficient, or whether it should use other means to ensure that the incompatibility will not arise. Note that an entity sending a message is notified about the delivery area, but not whether there was an entity within this area, so it does not know whether any entity actually received the message. Therefore, entities having received the message become responsible to ensure that no inconsistency arises with the entity that sent it, which corresponds to a *transfer of responsibility*. This transfer is however only partial (as the responsible entity remains responsible for the inconsistency in relation to other entities).

In our example, the responsibility for ensuring the safety constraints can be attributed to traffic lights. They will use both a transfer of responsibility and a delay on some of their actions. Every traffic light will send periodic messages to cars, informing them about its geographical position, its state, a time stamp for the message, and possibly a planned state change (i.e., a description of the change and the time at which it is planned to happen). Traffic lights will also delay actions that might raise an incompatibility (i.e., turning red) until they are sure that cars have been warned (and therefore that it is safe to do so).

4.2. Contracts

A responsible entity can use a combination of the three mechanisms mentioned 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, which summarises what entities can expect of each other. Contracts can include transfer of responsibility or not, and can include the possibility for an entity receiving a message from a responsible entity to give feedback.

In our example, every traffic light will send messages to cars when it is red, early enough so that they will have time to stop before the light. It will also warn cars when it turns to red early enough so that they have time to either pass through the light before it is red or stop. The specific parameters of this contract will be derived in the next paragraph.

4.3. Deriving the Parameters of the Contract

These contracts can be translated into geographical zones around entities. For example, the constraint that a traffic light will warn incoming cars that it is red early enough to allow them to stop, can be translated into a constraint on the zone in which communication must be guaranteed. More specifically, the traffic light needs to be able to communicate with cars in a zone that is big enough so that cars will have time to receive a message and stop before passing through the traffic light. This requires that it sends message at least over a zone of diameter $D_{present} + D_{period} + D_{O\_reaction}$, where $D_{duration}$ denotes the maximum distance travelled by a car during the time *duration*, *present* is the time required for an entity to become present once it has entered the actual coverage, *period* is the period of the traffic light messages, and $D_{O\_reaction}$ is the braking distance of cars.

If a message is not delivered in a big enough zone, the traffic light needs to prevent incompatibilities in another way, by adapting its behaviour: it should turn (or remain) green. Therefore, the traffic light should also send messages far enough so that it will have time to be notified if the message is not delivered, and react to it by switching to green if it was red, or cancelling its change to red if it was not. Hence, the traffic light needs to communicate over an area composed of two rectangles (one on each approach) of length:

$$\begin{aligned} CC &= D_{present} + D_{period} \\ &\quad + max(D_{O\_reaction}, D_{adaptNotif} + D_{R\_reaction}), \end{aligned} \tag{1}$$

where $adaptNotif$ is the time required for the traffic light to be notified of an adaptation, and $R\_reaction$ is the maximum time required by the traffic light to react to an adaptation notification (i.e., switching back to green, or canceling switching to red). This corresponds to the critical coverage as defined in the space-elastic model; if communication cannot be guaranteed in this area, the traffic light has to adapt its behaviour. The critical coverage is drawn with large dotted lines in Figure 2.

The delay with which the traffic light should warn cars that it will turn to red can be deduced similarly. Before actually turning to red, the traffic light has to ensure that all cars have received a message in time for them to stop. This requires that the actual coverage be bigger than the critical coverage. We will therefore distinguish two cases, depending on whether or not a message containing its intention to turn to red is delivered in an area bigger than the critical coverage (as calculated above) or not.

If the traffic light is notified that a message has been delivered in an area bigger that the critical coverage, it needs to ensure that the cars will have had time to receive the message and react before they arrive at the traffic light. If we assume that the car has a constant deceleration over its breaking distance, it will take $2 \times O\_reaction$ for the car to stop. Therefore, if we define Δ as the time required between the sending of the message and the time that the traffic light can turn to red, this implies:

$$\Delta \geq msgLatency + 2 \times O\_reaction\,. \tag{2}$$

If the message is delivered in an area smaller than the critical coverage, the traffic light has to change its plan and stay green. The traffic light should still have time to be informed of this change, and react to it (cancel the switching to red). This implies:

$$\Delta \geq msgLatency + adaptNotif + R\_reaction\,. \tag{3}$$

Δ represents the delay that the traffic light has to wait between announcing a switch to red and actually switching to red. Therefore, to meet the liveness requirements, Δ should be minimised. So from (2) and (3), we deduce:

$$\begin{aligned} \Delta = msgLatency + max(&2 \times O\_reaction, \\ &adaptNotif + R\_reaction)\,. \end{aligned}$$

If this condition is respected, when there is no critical adaptation before the event deadline all the cars present in the critical coverage will receive the message in time for them to stop; and when there is a critical adaptation the traffic light will be notified early enough so that it will not turn to red. Therefore, this condition is sufficient for the traffic light to switch safely to red.

4.4. Achievable Timing

From (1), the value of the critical coverage can be derived: for example, for a maximum car speed of 50 km/h, and a message period of 16 s, the critical coverage is around 250 m, which is within the transmission range of a typical wireless (e.g. IEEE 802.11b) transmitter. Therefore it is realistic to assume that timely communication can be assured within the critical coverage most of the time. Furthermore, with these values, it can also be derived that once red, the traffic light is assured to stay red for at least 15 s, which appears to be sufficient for pedestrians to cross (it is a value observed at traffic lights in Dublin).

5. Related Work

A number of attempts [3, 8] have been made to adapt the methods used in more reliable networks, typically group communication, to wireless networks. However, the guarantees offered are only probabilistic and are not suitable for safety-critical applications, furthermore, these do not offer timeliness guarantees.

Real-time systems typically rely on a synchronous model assumption (described for example in [14]), whose two main assumptions are that processing times and message-delivery delays are bounded. This model, however, is not realistic for wireless networks, where communication is highly unreliable. A more realistic model, called quasi-synchronous, has been suggested [14]. In this model, the assumption coverages are weakened, i.e., there is a known non-null probability that an assumption does not hold.

A protocol for ensuring real-time communication in dynamic wireless networks assuming a quasi-synchronous model has been proposed in [15]. In this work, communication is guaranteed with a varying latency. This paradigm is suitable for a class of applications, called *time-elastic*, which can execute in a degraded mode when communication cannot be guaranteed at the required latency. However, time-elastic applications are, by definition, not hard real-time.

6. Conclusion

In this paper, we introduced an alternative real-time communication paradigm for dynamic wireless networks. This model provides hard real-time guarantees over a varying geographical area. We demonstrated how to use the model to build reliable applications for dynamic wireless networks. The example presented might seem simplistic, but captures many crucial aspects, such as the presence of both mobile and fixed entities and a strong safety constraint that should be respected at all times. We studied the use of the space-elastic model for a variety of application scenarios. It generalises to scenarios including interactions between mobile components of different types. It appears that translating the safety constraints on requirements on entities and parameters for the space-elastic model is not trivial, and we are in the process of characterising a general model to do so, which could be supported by tools for application designers.

The authors are grateful to SFI for their support under Investigator award 02/IN1/I250.

References

[1] J. Cawkwell. A visually guided agv for use as passenger transport in urban areas. In *ITSC*, 2000.
[2] R. Cunningham and V. Cahill. Time bounded medium access control for ad hoc networks. In *POMC*, 2002.
[3] R. Friedman. Fuzzy group membership. In *Future Directions in Distributed Computing*, 2003.
[4] H. Hartenstein, B. Bochow, A. Ebner, M. Lott, M. Radimirsch, and D. Vollmer. Position-aware ad hoc wireless networks for inter-vehicle communications: the fleetnet project. In *MobiHoc*, 2001.
[5] S. Hirose and E. F. Fukushima. Development of mobile robots for rescue operations. *Adv. Robotics*, 16(6), 2002.
[6] B. Hughes, R. Meier, R. Cunningham, and V. Cahill. Towards real-time middleware for vehicular ad hoc networks. In *VANET*, 2004.
[7] H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. 1997.
[8] J. Luo, P. T. Eugster, and J.-P. Hubaux. Pilot: Probabilistic lightweight group communication system for ad hoc networks. *IEEE Trans. Mobile Comput.*, 3(2), 2004.
[9] R. Meier and V. Cahill. Exploiting proximity in event-based middleware for collaborative mobile applications. In *DAIS*, volume 2893 of *LNCS*, 2003.
[10] R. Meier, B. Hughes, R. Cunningham, and V. Cahill. Towards real-time middleware for applications of vehicular ad hoc networks. In *DAIS*, volume 3543 of *LNCS*, 2005.
[11] E. Nett and S. Schemmer. Reliable real-time communication in cooperative mobile applications. *IEEE Trans. Comput.*, 52(2), 2003.

[12] E. Nett and S. Schemmer. An architecture to support cooperating mobile embedded systems. In *Computing Frontiers*, 2004.

[13] R. D. Schraft. Mechatronics and robotics for service applications. *IEEE Robot. Automat. Mag.*, 1(4), 1994.

[14] P. Verissimo and C. Almeida. Quasi-synchronism: a step away from the traditional fault-tolerant real-time system models. *IEEE TCOS Bulletin*, 7(4), 1995.

[15] P. Verissimo and C. Almeida. The timely computing base model and architecture. *IEEE Trans. Comput.*, 51(8), 2002.