We don't need to agree to coordinate

Mélanie Bouroche and Vinny Cahill
Distributed Systems Group
Trinity College Dublin

Abstract

Autonomous mobile entities are playing an ever increasing role in our everyday lives. When such entities operate in the same environment, they need to coordinate their behaviour in order to ensure system-wide safety constraints. As these entities are mobile, they can only communicate over wireless (possibly ad hoc) networks. In wireless networks, however, communication is unreliable, therefore consensus-based coordination cannot be applied. This means that entities need to take decisions independently, while having access to only limited information.

This short paper describes an approach for entities to coordinate their behaviour without needing to reach a consensus. This approach is based on the notion of responsibility, which allows the duty of ensuring the safety constraints to be distributed over entities. We define the responsibility condition formally, and give an intuition as to how it can be used to by autonomous entities to coordinate their behaviour.

1 Introduction

Progress in the miniaturisation of computing devices, wireless communication, and sensing technology are encouraging the deployment of autonomous mobile entities in our everyday environment. Examples of such entities include automated guided vehicles [1] and mobile service robots [7]. To ensure safe operation, such entities need to coordinate their behaviour in real-time both with each other and with their environment in order to ensure that system-wide safety constraints are respected.

Entities typically coordinate their behaviour by agreeing on a set of actions or on a common view of their environment. As these entities are mobile, however, they can only communicate over wireless (possibly ad hoc) networks, where communication, and in particular real-time communication is highly unreliable and the achievable performance varies greatly over time and location [3]. Distributed consensus, however, is not solvable in the presence of an arbitrary number of communication failures [6]. Therefore consensus-based coordination approaches cannot be applied and entities need to take decisions independently, while having access to only limited information.

Our approach relies on the observation that often, entities do not need to agree on their view of the world or their actions to ensure safety constraints, but that instead some entities can take responsibility for ensuring them independently. For example, an entity could delay taking an action that might violate the safety constraint. Unless the coordination problem is trivial, ensuring the safety constraints must hinder the progress of such entities, for example, by delaying a desired action. The coordination problem then becomes a problem of how to ensure that, at any time, in every group of entities whose states might violate a safety constraint, at least one entity ensures that they do not. We call this entity a responsible entity, and this condition the responsibility condition.

This paper is organised as follows: Section 2 formalises the coordination problem, and Section 3 defines the notions of responsibility and entity compatibility, and uses these notions to define the responsibility condition. Finally Section 4 discusses how the responsibility condition can be used to coordinate the behaviour of entities, and how it can be fulfilled. Section 5 concludes the paper.

2 Model and the Coordination Problem

In this section, we define the system model and formalise the coordination problem. Safety constraints typically include constraints on the actions of entities and on their state. One of the particularly relevant parameters for mobile entities is the relative positions of entities, and in particular the distance between them. For example, an emergency vehicle only needs to coordinate with cars that are on the same stretch of road. This section introduces a formalism to express safety constraints and to define exactly which states of a set of entities violate the stated constraints.

2.1 System model

The system model consists of a finite but a priori unknown collection of autonomous mobile entities operating in the same environment. These entities coordinate their behaviour by communicating over a wireless network. Messages can be lost or delayed. We do not, however, consider the possibility of the entities themselves failing.

2.2 Scenario, modes and states

A *scenario* encompasses a set of *entities* of types $E_1, E_2, .., E_n$ and a *safety constraint*. For example, when considering the scenario of an emergency-vehicle warning system, the entity types might represent cars and emergency vehicles. A possible safety constraint for this scenario may be that no emergency vehicle should collide with any car (for simplicity, interactions between cars and between emergency vehicles are not considered in this scenario but can be modelled using the same techniques). In this definition, a scenario has a single safety constraint; if the interaction of the entities of a scenario must fulfil several safety constraints, either several scenarios can be defined, or these safety constraints can be linked using a logical conjunction.

The behaviour of an entity depends on its type, and is defined by a set of modes of operation, termed simply *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, given the maximum speed of an emergency vehicle $v_{\max}$, and an increasing set of speeds $\{v_i\}_{i\in[1,p]}$ with $v_p = v_{\max}$, the modes of emergency vehicles can be defined as: `stopped`, $\{$`going_at_v`$_{\mathtt{i}}$, `accelerating_to_v`$_{\mathtt{i}}$, `braking_to_v`$_{\mathtt{i-1}}\}_{i\in[1,p]}$. Similarly, the behaviour of a car can be modelled with the modes `travelling`, `getting_out_of_the_way`, and `out_of_the_way`.

The situation of an entity at a given time is described by its *state* which is made up of a number of *state variables*: the entity's mode, position and possibly some application-specific information. The (current) mode of an entity describes the action(s) that this entity is currently undertaking, while an entity's state describes its current situation, including its position. Modes can therefore be used to predict state evolution. For example, the state variable "position" of a car will remain constant as long as it is in the mode `stopped`. In the example, the states of both emergency vehicles and cars can be defined as combination of their mode, location, current speed and direction.

2.3 State compatibility

The states $(s_{e_1}, s_{e_2}, ..., s_{e_m})$ of a finite set of entities $\{e_1, e_2, ..., e_m\}$ are said to be *compatible* if the safety constraint is not violated when the entities are simultaneously in these states. This relation is denoted as $\mathcal{C}_{\mathrm{S}}(s_{e_1}, s_{e_2}, ..., s_{e_m})$. For example, the states of an emergency vehicle and a car are compatible if they are far enough away. Also, the state of a car that is off the road is compatible with the state of any emergency vehicle.

2.4 Expressing the safety constraints

The safety constraints can be expressed as a conjunction of incompatibilities between states, including constraints on the relative distance of entities (denoted distance($position1, position2$)). For example, the safety constraint that no emergency vehicle should collide with any car can be stated as: for any state s_{car} of any entity of type "car" and any state s_{ev} of any entity of type "emergency vehicle",

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

This expresses the fact that the safety constraints will not be violated if a car and an emergency vehicle are further than d apart, or the emergency vehicle is stopped, or the car is out of the way of the emergency vehicle.

Using this high-level and implementation-independent formalism, the coordination problem can be expressed as, at any time, the states of all the entities of a scenario must be compatible.

3 The Responsibility Condition

This section presents the responsibility condition, the core of our approach to coordination. First the notion of responsibility is introduced, then entity compatibility is defined, and is used to define the responsibility condition.

3.1 Responsibility

Our approach to coordination relies on the idea that for every set of entities whose states might violate the safety constraints, at least one of the entities needs to ensure that it does not occur. We say that this entity is *responsible* for ensuring the safety constraint.

Note that the responsibility for the different incompatibilities that make up the safety constraint can actually be attributed to different entities, but to simplify the explanation we will assume that responsible entities are responsible for the entire safety constraint in the following.

3.2 Entity compatibility

A set of entities $\{e_1, e_2, ..., e_m\}$ are compatible, denoted $\mathcal{C}_{\text{e}}(e_1, e_2, ..., e_m)$, if all states $\{s_1, s_2, ..., s_m\}$ in which these entities can be are compatible. For example, the states of two or more entities of type car will never violate the safety constraint stating that cars and emergency vehicles should not collide, so these entities are compatible. (If all entities are compatible, the coordination is trivial.)

3.3 Responsibility condition

The responsibility condition states that for any set of entities whose state might violate the safety constraint, at any time, at least one entity must be responsible (to ensure that it does not happen). If we define the predicate

$$\mathcal{R}(e, t) : \text{entity } e \text{ is responsible at time } t\,,$$

then the responsibility condition can be expressed as

$$\forall m, \forall e_1, e_2, ..., e_m, \neg\mathcal{C}_{\text{e}}(e_1, e_2, ..., e_m) \Rightarrow \forall t\, |\{e_i : \mathcal{R}(e_i, t)| \geq 1\,.$$

This approach only caters for a class of applications for which some entities can, independently of other entities, take some actions to ensure that the safety constraints will not be violated. While this criteria restricts the domain of application, we found that many applications fit into this category. For example, for many mobile entities, it might be sufficient to stop to ensure that safety constraints are not violated. Similarly, it is sufficient for a pedestrian traffic light to remain green to ensure that no cars will go through a red light (though during that time, pedestrians will not be able to cross the road).

4 Using the responsibility condition

This section first discusses how the responsibility condition can be used for the coordination of autonomous mobile entities, and then how it it can be fulfilled.

4.1 Coordinating using the responsibility condition

While the responsibility condition holds, there will be at least one responsible entity in every set of entities whose states might violate the safety constraint. To cater for the possibility of having several responsible entities, the way in which responsible entities guarantee that the safety constraints are not violated must be conservative, i.e., several entities must be able to take this action simultaneously. Consider, for example, that a car stops before entering a four way junction to let vehicles from another direction pass through the junction. If the vehicles approaching from all directions adopt the same strategy, the safety constraint that only cars coming from one direction should cross the junction simultaneously will still be ensured (though no car will progress through the junction). Similarly, if two emergency vehicles send messages to warn vehicles to get out of their way, they might both receive messages from one another, and get out of each other's way, which will be safe.

As discussed above, responsible entities must be able to ensure that the safety constraints will not be violated. This will depend on the action they are taking, or, as defined in 2.2, their mode. For responsible entities to make progress, they need to take actions that could potentially violate the safety constraints. They can only take such actions when they can communicate with other entities to warn them, which corresponds to a transfer of responsibility. This requires entities to have access to some feedback on the currently available state of communication. Such feedback can be provided by mechanisms such as collision detectors [2], but also communication models such as the space-elastic model [5].

4.2 Fulfilling the responsibility condition

Ensuring the responsibility condition is similar to the classical distributed systems leader election problem [4], except that instead of aiming to have at most one leader, the problem is to have at least one responsible entity. Intuitively, this problem is easier as when an entity A is transferring responsibility to an entity B, entity A can remain responsible until it finds out that B has received the message and is responsible.

The *role* of an entity is defined with respect to an interaction, as in object-oriented software engineering [8]. By default, entities only have a single (default) role, but if a safety constraint refers to several entities of the same type, they are distinguished by their role. So, for example, if the safety constraint in a scenario with autonomous cars states that two cars must not collide, then cars might be distinguished depending on their role: 'following' or 'leading', which depend on their relative positions.

Our approach assumes an initial partition of responsible entities so that, initially, the responsibility condition holds. Responsibility can be attributed initially to entities of certain types or to entities in certain roles. For example, emergency vehicles might be responsible to ensure that they do not collide with cars or cars might be responsible for ensuring that no other car collides into them from behind, so cars in the leading car role are responsible for possible state incompatibilities with cars behind them.

The responsibility condition is then maintained by having responsible entities transfer their responsibility to other entities before they take actions that could violate the safety constraint.

5 Conclusion

In this paper, we defined the responsibility condition, an alternative to consensus for the coordination of autonomous entities. We then gave an intuition about how autonomous entities can coordinate their behaviour using the responsibility condition. Note that, while it requires that entities have feedback about the current state of communication, and that in these conditions consensus might be solvable, there might be communication models in which the responsibility condition is solvable, while consensus is not. In addition, and more importantly, because with the responsibility condition communication is not required to ensure the safety constraint, but only to make progress, it is particularly suitable for coordination with real-time requirements, as is often exhibited by autonomous mobile entities.

References

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

[2] Gregory Chockler, Murat Demirbas, Seth Gilbert, Calvin Newport, and Tina Nolte. Consensus and collision detectors in wireless ad hoc networks. In *Proceedings of the annual ACM symposium on Principles of distributed computing (PODC'05)*, pages 197–206. ACM Press, July 2005.

[3] 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.

[4] Hector Garcia-Molina. Elections in a distributed computing system. *IEEE Transactions on Computers*, 31(1):48–59, January 1982.

[5] Barbara Hughes. *Hard Real-Time Communication for Mobile Ad Hoc Networks.* PhD thesis, Dept. of Computer Science, Trinity College Dublin, October 2006.

[6] Nancy A. Lynch. *Distributed Algorithms.* Morgan Kaufmann, 1996.

[7] Ioannis M. Rekleitis, Gregory Dudek, and Evangelos E. Milios. Multi-robot collaboration for robust exploration. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):7–40, October 2001.

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