# Membership Service Specifications for Safety-Critical Geocast in Vehicular Networks

Marco Slot, Mélanie Bouroche, Vinny Cahill
Trinity College Dublin
School of Computer Science and Statistics
Distributed Systems Group
{slotm,melanie.bouroche,vinny.cahill}@scss.tcd.ie

***Abstract*—Geographic group communication is a promising technique for collaborative driving applications. While one-way, geographic broadcast (geocast) is well-studied in vehicular networks, there has been little work to address the challenging reliability problems that arise. Not only do vehicular networks experience highly variable packet loss, but communication failures are difficult to detect. The presence of vehicles in an area and thus the set of vehicles that is supposed to receive a geocast, changes continuously. Without an expectation of the vehicles that should respond to queries, communication failure cannot easily be distinguished from absence. This requires a cross-layer approach, exploiting communication, sensing and driving rules. In this paper, we propose a membership service that provides group views for geographic areas as a building block to reliable geocast. We specify the semantics of the service and discuss different ways of implementing it. Finally, we show how it can be used for safe, collaborative driving.**

***Index Terms*—Inter-Vehicle Communication, Reliable Broadcast, Membership Service, Autonomous Driving, Vehicle Safety**

## I. Motivation

Intelligent Transportation Systems require new communication paradigms to enable sophisticated driving applications. Current efforts in vehicular networking focus primarily on driver support and notification applications [6], [7]. As vehicles become more autonomous, communication will play an increasingly important role in motion control and collaborative driving. This novel application domain requires reliable communication primitives, which allow a sender to determine whether its messages arrived [8]. Sufficient feedback is necessary to be able to rely on the message outcome (e.g. changed behaviour), but is difficult to achieve in a vehicular network.

Vehicular networks have several distinctive features that make communication inherently unreliable. Radio communication in a dynamic environment with a rapidly changing topology has highly variable reliability as radio signals can be obstructed and network partitions occur. Since driving applications have hard real-time constraints [8], guarantees of eventual success are not applicable. With an unreliable channel, acknowledgement is required to confirm reception. Unfortunately, it can also be very difficult to detect communication failures. Driving applications will use geographic broadcast (geocast) to send messages to an area of relevance [7], rather than a specific vehicle. However, without an expectation of which vehicles should respond, it is impossible for a sender to distinguish between a vehicle that fails to acknowledge its messages and one that is not in the area at all.

Sender-initiated, reliable group communication protocols use a membership service to determine which nodes are potential receivers [1], [8]. The service generates a current *view* of the broadcast group. A sender detects a communication failure if a group member does not respond to a broadcast. To detect communication failures for inter-vehicle geocast, we need to build a reliable view for the target area at a given time. Current work in reliable broadcast for vehicular networks relies on the ability of vehicles to eventually make themselves known through communication [4], [5], [3]. For truly safe, collaborative driving to be realized, applications will need to be prepared to deal with *any* communication failure.

In this paper, we specify a membership service for geographic, inter-vehicle group communication. Our approach relies on sensors and driving rules to reliably tie the state of communication to the physical state of the system. While driving applications are likely to have many sensing and tracking systems already, our goal is to extract the information that is relevant to the state of communication and provide strong guarantees. Safety-critical applications will need to have verifiable properties that we make explicit by providing them through a separate membership service. We will further show that implementations of a service providing only membership information can be relatively simple and reliable compared to more sophisticated sensing systems.

The goal of membership is to confirm the absence of vehicles that fail to communicate in a target area. The membership abstraction we propose, specifies which communicating vehicles *could be* in an area at a given time, based on the information that is available at an observer. Most important is the guarantee that membership provides: no other vehicles exist in the area. If a geocast from the observer reaches all the vehicles in the membership set (members), it is ensured that it reaches every vehicle in the area.

We begin by specifying the semantics of membership information in Section II. We define interfaces for a membership service and a reliable broadcast service in Sections III and IV. In Section V, we discuss different ways in which membership could be implemented. Finally, we demonstrate how membership can be used to develop a safe freeway driving application in Section VI. Section VII concludes the paper.

## II. Membership Information Specification

Membership is an abstraction of presence information, across communication, sensing and driving rules. To formalize and clarify its exact meaning we define it as a logical relationship, seen in Formula 1. A membership$(M, A, t)$ tuple specifies that there are no other vehicles than those listed in $M$ in area $A$ at time $t$. This rule is a safety property that must at all times be met by the implementation of the service. Membership tuples exist in the internal state of a vehicle and are determined by the information available at that vehicle under conservative assumptions.

$$\text{membership}(M, A, t) \Rightarrow \forall v \in V : \text{in}(v, A, t) \Rightarrow v \in M \quad (1)$$

$V$ : Set of all vehicles

$M \subseteq V$ : Set of members

$A$ : Area of interest (membership area)

$t$ : Point in time

$\text{in}(i, A, t)$ : Vehicle $i$ is in area $A$ at time $t$

The membership guarantee allows the sender of a broadcast to conclude that, if communication succeeds to all members $M$, all vehicles that were in area $A$ at time $t$ were reached. In the event of a communication failure, it can learn which of the members have not responded and take appropriate action (e.g. retry). We require all potential members to be *compatible*, meaning they are equipped to respond and follow the membership protocols. There will not be any membership tuples for an area which contains incompatible vehicles, since it is not possible to communicate with them.

Membership does not necessarily imply that the members are actually in the area. There may not be sufficient information to definitively conclude that a vehicle is outside of the area. Since the guarantee must be maintained, this will result in a larger membership set or a reduced area. One limiting factor on the quality of the information is its *decay* over time. The time in a membership tuple can be shifted forward, effectively deriving future membership information from the present. Vehicles that are near the borders of the membership area when the information was obtained may have left by the time it is used. Without newer information, it cannot be concluded that those vehicles are now outside of the area. They must be included in the membership set to ensure the guarantee is met. Another type of decay relates to unknown vehicles entering the area after the information was recorded. The area shrinks according to the bounds given by the driving rules of the system (e.g. maximum velocity) as time progresses. A brief example is displayed in Figure 1. The result of decay is that the membership set remains the same while the area shrinks.

Computing decay is necessary to reason about the current environment using older information, but it requires knowledge of the dynamics of the system. We expect there to be a service that can be called by a local membership library to get the necessary information on the dynamics to compute decay. Applications can call the same service to provide new instructions on higher level rules. We will not precisely specify such a service in this paper, but will provide various examples.

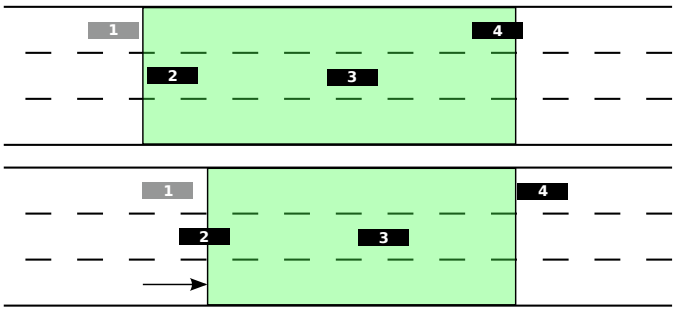


Fig. 1. Membership decay from membership$(\{v_2, v_3, v_4\}, A, 0)$ to membership$(\{v_2, v_3, v_4\}, decay(A, 1), 1)$. As time progresses the area in which absence of unknown vehicles can be confirmed shrinks. At $t = 1$, only one member is still fully in the area. New information is required to increase the area or reduce the membership set.

If multiple membership tuples that overlap are available, they can be merged into a single one. The advantage of merging tuples over considering them seperately is that it expands the borders from which decay of the area occurs. It is also easier to reason about a single tuple. To merge tuples, they first need to be shifted (decayed) to the same point in time. After that, the merged tuple is simply the union of the areas and the membership sets.

The application works directly with the membership tuples and the guarantees they provide. In the ideal case, there is only one tuple, its area is exactly the size of the desired area and its member set consists only of the vehicles that are physically present in that area. In reality, membership information may only be available for a much larger area and a much larger set, or smaller sub-regions of the desired area. Whether this is sufficient depends on the application, which requires some logic to adapt to it.

## III. Membership Service

A membership service gathers membership tuples for a specific area. To achieve this, communication is required to confirm the members are compatible and obtain information that is gathered remotely. Sensing is required to match members to physical locations. Knowledge of driving rules is required to reason about membership decay. How tuples are generated from these three abilities is up to the implementation and discussed in Section V. Here we discuss a possible interface for the service:

(Input) $\mathbf{gather}(area, t_{deadline})$

(Output) $\mathbf{view\_change}(\mathcal{M})$

$area$ : The area for which to gather membership tuples

$t_{deadline}$ : The time for which to gather membership

$\mathcal{M}$ : A set of membership tuples

The service is activated by an $\mathbf{gather}(area, t_{deadline})$ input. From that moment until $t_{deadline}$, the membership service will generate $\mathbf{view\_change}(\mathcal{M})$ outputs whenever new membership tuples are available. $\mathcal{M}$ is a set of membership

tuples, of which the areas overlap with the gathering area. If no membership information is available a $\mathbf{view\_change}(\emptyset)$ is generated. The tuples should be provided in their smallest (unmerged) form to reveal all available information. Although information could be available sooner, the goal of the membership service is to have the information at $t_{deadline}$. This knowledge allows it to anticipate the decay of information. After $t_{deadline}$ has passed no new **view_change** events will be generated.

## IV. Reliable Broadcast Service

A reliable broadcast guarantees eventual delivery of all the data to all the group members [2]. For geographic broadcast in vehicular networks, this definition needs to be nuanced. Applications have real-time constraints and vehicles may permanently leave the target area. We cannot guarantee eventual delivery to group members in those cases. However, as long as communication failure can be detected and there is non-zero probability of success, the vehicle can adapt and retry with a different area or group, eventually achieving success. This eventual guarantee depends on the application. For example, if a vehicle on a freeway fails to realize a lane change in one area due to communication failure, it will have to retry further down the road in a different area with potentially different vehicles. We will use the term reliable purely to refer to the ability of the *sender* to detect communication failures.

The goal of the broadcast service is that all vehicles that are or could be in a given area at a given time receive the message. Every receiver should send an acknowledgement back to the sender. If any communication failure occurs the sender can detect it by comparing the list of vehicles from which it receives an acknowledgement against the membership set for the area. Adapting to the combination of communication failures and failure to obtain membership may require complex application-specific logic. The reliable broadcast service will only act as a thin wrapper over lower layer communication and membership services and pass on the information. The interface for the reliable broadcast service is as follows:

(Input) $\mathbf{broadcast}(handle, sender, msg, area, deadline)$

(Output) $\mathbf{result}(handle, R, \mathcal{M})$

$handle$ : Logical handle that maps a result to a broadcast

$R$ : Vehicles that acknowledged the broadcast

When a **broadcast** event is generated by the application the service generates an $\mathbf{gather}(area, deadline)$ event at the membership service. At this point the service can proceed in two different ways:

- *Geocast*:
  The broadcast service directly performs a geographic broadcast to the area, with the size of the area adapted to consider decay.
- *Multicast*:
  The broadcast service waits for a **view_change** event from the membership service to obtain a set of members for the target area. It can then perform a group multicast to the set of members.

In both cases, the result is the set of vehicles $R$ that acknowledged the message. If this is a superset of the set of members for the target area then communication succeeded. Alternatively, the result can be mapped to the part of the area in which communication did succeed by considering only those membership$(M, A, t)$ tuples for which $M \subseteq R$.

The two communication approaches have different properties. The second approach likely causes more overhead as packets need to be routed to specific nodes rather than a contiguous area. On the other hand, it is a more informed approach that takes into account members that may be outside the area. Which will work best in part depends on the implementation of the membership service. The geocast can be quite efficient if the membership information is obtained from the communication itself. The multicast approach will work much better in cases where many members may be outside of the area. The implementation examples we give in the next section reflect these differences. A realistic approach will likely use a combination of methods to achieve a high quality of service.

## V. Implementing the Membership Service

Implementing the membership guarantees will be quite challenging. Due to its all or nothing character, radio communication is not a reliable means of detecting presence. Sensors degrade more gracefully, providing at least a basic level of service and sophisticated error detection. Guarantees from autonomous vehicles that they will obey certain traffic rules can also provide reliable presence information. In this section, we give two very different examples of how membership can be achieved using these principles.

### A. Example 1: Radar-based membership

An increasing number of cars are equipped with radars to detect other vehicles. Although still limited to certain angles, we expect this to grow to full 360° views in the future. This sensor then allows us to read the presence or absence of vehicles in the surrounding area of a communicating vehicle. Recognizing vehicles from radar data is error-prone. Even if edges on the radar can be identified as a vehicle, vehicles may still shadow other vehicles. Fortunately, recognizing vehicles is not really necessary for membership. The main goal is to confirm absence of vehicles that do not communicate. To do so, all that is really needed is to confirm that the space surrounding the members is empty. Conveniently, the members can use their own radars to achieve this.

When a $\mathbf{gather}(area, t_{deadline})$ event is triggered, the membership service issues a geographic broadcast to an area that is slightly larger than the requested $area$. The additional space is needed to take into account the decay up until $t_{deadline}$. At this point, the service needs knowledge of the driving rules to determine how the area will decay. In the freeway example we will see later, it should extend the head of the area by $v_{max}.(t_{deadline} - t_{current})$ to account for vehicles driving towards the area at maximum speed, while the tail of the area can be left unchanged since vehicles may not enter

it going backwards. Every vehicle that receives the query, generates a 2-dimensional polygon of the area that appears empty on its radar after noise removal. The polygon connects the end points of distance measurements or, when no distance was measured, the maximum range of the sensor. To factor out GPS errors, the distances can be reduced by the error margin of the GPS device. The area taken up by the vehicle itself and its direct surroundings are always included in the polygon. The vehicle sends back its position, the GPS time and the points of the polygon relative to its own position to the requester. The membership service will generally be used in combination with a broadcast to more or less the same area. In this case, the implementation can reduce overhead by attaching the membership query and response to the broadcast packets of the user and the resulting acknowledgements.

When the membership service receives a polygon $A$ from a vehicle $m$ at time $t$ it translates the polygon to its local representation $A'$. It then generates a $\mathbf{view\_change}(\mathcal{M} \cup \{\text{membership}(m, A', t)\})$ event. According to our specifications, the service itself returns the raw tuples as to not hide any information. Although the membership service does not merge the data, it is still important to understand what happens in this case. Other vehicles will appear as gaps in the polygon of a radar image. These gaps are filled when the polygon is overlapped with the data from the member itself. The gaps will grow rapidly under decay. It is important that the radar snapshots are gathered with minimal time differences to be able to merge them without decay.

Decay is computed by using the worst-case bounds of the driving rules for unknown vehicles that could be just outside the polygon, similar to Figure 1. Since the edge of the polygon may be determined by the presence of a vehicle at that location this is not overly conservative. One problem that arises is small gaps formed by merging polygons. If the gap is big enough to fit a small car or even motorbike, decay occurs from the gap as much as it does from the edge.

### *B. Example 2: Access Control Membership*

The previous example relied only on the most basic driving rules. If rules existed that excluded the possibility of unknown vehicles being present, achieving membership could be more straightforward. Vehicles regularly need to travel through toll gates that seal off access until a fee is paid. A similar approach can be applied to membership. If a membership authority exists, access gates can be put in place that remain sealed until the vehicle is registered with the authority. Here we require all vehicles that wish to enter the road to be compatible, but the required equipment is no more than what is already provided by modern mobile phones. Registration must go through some well-known, static access point to ensure vehicles can register even if there are no other vehicles. We assume a centralized system to keep track of registration and provide membership information for the entire road. The vehicles need to be able to communicate with this system at all times to obtain membership information. They can avail of the mobile telephony network or Wifi access points along the road.

The membership authority is in charge of generating the membership$(M, A, t)$ tuples. By definition, it can generate a tuple for the entire road, containing all members. Clearly, this is not very useful as vehicles would have to contact every vehicle on the road to communicate with the desired area. To provide more fine-grained tuples the authority needs to keep track of where the vehicles are located. The local membership service at each vehicle must periodically provide a position update to the authority, obtained from GPS. The service forwards its $\mathbf{gather}(area, t_{deadline})$ to the authority, which returns membership$(M, area, t_{deadline})$. $M$ is the union of set of vehicles whose last-known position was in the area, and those that could enter it before $t_{deadline}$. If the vehicle fails to query the authority it simply does not have membership information, to which it should adapt.

Since there is global knowledge of the road, decay can be made more optimistic than in the previous example. Along with a membership tuple, the authority can send a complete list of vehicles that are driving towards the membership area. The local membership service can then compute when the closest vehicle in the list will enter the area. Based on that information it can either reduce the area, or add the vehicle to the membership set without changing the area. A global, worst-case decay still occurs from the furthest vehicle in the list, as unknown vehicles may exist beyond that.

The system described here is not very robust against vehicles that stop communicating after successfully entering the system. The position of these vehicles would only be known at the start and would eventually have to be added to all membership sets. Since the vehicle is not communicating, none of the reliable broadcasts will succeed. To solve the problem and to also allow incompatible vehicles into the system there could be an additional sensor-based tracking mechanism. One possibility is that the cameras that are used for enforcing speed limits on freeways today could also be used to keep track of the position of a vehicle by its number plate. This position information would allow the membership authority to compute which area the non-communicating vehicles could be in. This area then needs to be excluded from membership tuples.

## VI. Lane Merging through Membership

In this section, we demonstrate how the membership service can be used to support a safety-critical freeway driving application. Specific collaborative driving algorithms and protocols are beyond the scope of this paper. Instead we define a simple logical model for states and safety constraints and show how membership can be used to reason about state changes. For state information we define a $state$ tuple:

$\text{state}_i(L_i, l_{i\ target}, p_i, v_i, t_i)$.

$L_i \subseteq L_{all} = \{1, 2, ..., n\}; l_{i\ target} \in L_{all} \cup \{\perp\}; p_i, v_i, t_i \in \mathbb{R}$.

Each vehicle has a unique identifier, represented by $i$. The 1-dimensional position and velocity of the vehicle on the lane are defined in $p_i$ and $v_i$. $L_i$ is the set of lanes the vehicle is currently on, at least one lane and at most two adjacent lanes. A vehicle may only change to the lane to which its $l_{i\ target}$ is

set. The target lane intention determines what responsibilities the vehicle needs to take on with regard to other vehicles. Finally, each vehicle has an internal clock $t_i$ synchronized to GPS time.

We define a set of constraints on the state that will ensure the safety of the system. Constraints on velocity and target lane can be trivially applied. Less straightforward are constraints on the minimum distance to other vehicles. This requires knowledge of where those vehicles are and should consider vehicles on other lanes. For vehicles that are driving forward, we only apply a distance constraint to vehicles ahead of it ($p_i < p_j$) on the same lane(s) ($L_i \cap L_j$). While $l_{i\ target}$ is not $\perp$, the vehicle also needs to maintain distance to those behind it, and those that may be on the target lane or intend to go there. We formally define these constraints as follows:

$\forall$ state$_i(L_i, l_{i\ target}, p_i, v_i, t_i)$ :

- Do not travel backwards or break the speed limit: $v_i \geq 0; v_i \leq v_{max}$.
- Only target adjacent lanes: $l_{i\ target} \neq \perp \Rightarrow \forall k \in L_i : |l_{i\ target} - k| \leq 1$.
- Keep distance when driving:
  $\forall$ state$_j(L_j, l_{j\ target}, p_j, v_j, t_j)$ :
  $p_i < p_j \wedge L_i \cap L_j$
  $\Rightarrow p_j - p_i \geq d_{min}(v_i)$.
- Keep distance when merging:
  $\forall$ state$_j(L_j, l_{j\ target}, p_j, v_j, t_j)$ :
  $l_{i\ target} \neq \perp \wedge$
  $(L_i \cap L_j \vee l_{i\ target} \in L_b \vee l_{i\ target} = l_{j\ target})$
  $\Rightarrow |p_i - p_j| \geq d_{min}(v_j)$.
  $d_{min}(v)$ : Minimum distance at velocity $v$

If a vehicle wishes to proceed to a new goal state state$_{goal}(L_{goal}, l_{goal\ target}, p_{goal}, v_{goal}, t_{goal})$ it needs to verify whether it and the transition to it is safe. For a goal state that is in front of the vehicle, it needs the distance to the vehicle(s) ahead of it to meet the driving constraint. If the new state is on another lane, the vehicle will first need to go into a transitional state in which $l_{target}$ is set ($state_{merge}$). This activates the merging constraints, which needs to be checked against all other vehicles that may be on that lane by $t_{goal}$. Only a few of those vehicles can actually come close enough to the goal state before it is reached to break any constraints. The application can compute the size of the area in which these vehicles may be. We call this area the conflict zone $C$ and define the $in$ function for it:

$p_{conflict\ start} = p_{merge} - d_{min}(v_{max}) - v_{max}.(t_{goal} - t_{merge})$.

$p_{conflict\ end} = p_{goal} + d_{min}(v_{goal})$.

$C = [p_{conflict\ start}, p_{conflict\ end}]$.

$\text{in}(i, C, t) \Leftrightarrow p_{conflict\ start} \leq p_i$ at time $t \leq p_{conflict\ end}$.

The application can use the reliable broadcast service to contact the vehicles in the conflict zone by generating a $\mathbf{broadcast}(handle, id, message, C, t_{goal})$ event. The message is a request to vehicles for their trajectories and the promise that they will not change it until $t_{goal}$. The sender waits for responses from each of the vehicles and the $\mathbf{result}(handle, R, \mathcal{M})$ event. From the result $\mathcal{M}$, it attempts to construct a membership$(M, C, t_{goal})$ tuple through merging and decay. If it fails to do so it needs to change to a more conservative goal state and retry later. If there is sufficient membership information, then it is guaranteed that $\forall b \notin M \Rightarrow \neg in(b, C, t_{goal})$. This means that the safety constraints will not be broken for any vehicles not in $M$.

From every vehicle that is in $M$, the application needs to either receive a trajectory confirming that it will not conflict with the goal or the transitional states, or a position that confirms the vehicle will not be in $C$ at that time. In this case, it is not sufficient for a member to be in $R$, since that does not allow the vehicle to predict where the vehicle will be at $t_{goal}$. If all members responded and none of the trajectories conflict, the merge may proceed. Otherwise, the vehicle has until $t_{merge}$ to adapt by retrying to query selected vehicles or adapting the goal state.

## VII. Conclusion

A membership service is a building block for safe, collaborative driving applications. It provides rich feedback on communication results by mapping them to the physical state of the system. We have shown how a membership service can be used to fulfill the safety constraints of a basic driving application. In the future, we will evaluate the implementation techniques we discussed and research how a high quality of service can be achieved.

## VIII. Acknowledgement

The work described in this paper was partly supported by Science Foundation Ireland (as Research Frontiers award 08/RFP/CMS1470 and Lero - The Irish Software Engineering Research Centre) and by the Irish Higher Education Authority Programme for Research in Third Level Institutions (as the Networked Embedded Systems Centre).

## References

[1] G. V. Chockler, I. Keidar, and R. Vitenberg. Group communication specifications: a comprehensive study. *ACM Computing Surveys*, 33(4):427–469, December 2001.

[2] S. Floyd, V. Jacobson, C.-G. Liu, S. Mccanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5:784 – 803, December 1997.

[3] M. Li, W. Lou, and K. Zeng. OppCast: Opportunistic broadcast of warning messages in vanets with unreliable links. *IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 534 –543, October 2009.

[4] N. F. Maxemchuk, P. Tientrakool, and T. L. Willke. Reliable neighborcast. *IEEE Transactions on Vehicular Technology*, 56(6):3278–3288, November 2007.

[5] F. Ros, P. Ruiz, and I. Stojmenovic. Reliable and efficient broadcasting in vehicular ad hoc networks. *IEEE Vehicular Technology Conference (VTC Spring)*, pages 1 –5, April 2009.

[6] M. L. Sichitiu and M. Kihl. Inter-vehicle communication systems: a survey. *IEEE Communications Surveys & Tutorials*, 10(2):88–105, July 2008.

[7] Y. Toor, P. Muhlethaler, and A. Laouiti. Vehicle ad hoc networks: applications and related technical issues. *IEEE Communications Surveys & Tutorials*, 10(3):74–88, September 2008.

[8] T. L. Willke, P. Tientrakool, and N. F. Maxemchuk. A survey of inter-vehicle communication protocols and their applications. *IEEE Communications Surveys & Tutorials*, 11(2):3–20, June 2009.