Specification, Implementation and Verification of Dynamic Group Membership for Vehicle Coordination

Mikael Asplund*, Jakob Lövhall* and Emilia Villani†
*Department of Computer and Information Science
Linköping University
Sweden
Email: mikael.asplund@liu.se
†Instituto Tecnolgico de Aeronutica - ITA
So Jos dos Campos SP
Brazil
evillani@ita.br

***Abstract*—New advanced traffic management solutions with fully or semi-autonomous vehicles that communicate over a wireless interface to coordinate their driving decisions create new challenges in distributed computing. In this paper we address the problem of dynamic group membership in three stages. First, we propose three criteria to specify correctness and performance of the group views created by such algorithms in terms of soundness, completeness and freshness. Second, we develop a group membership protocol tailored for vehicular coordination. Finally, we show through simulation and model-based verification that the protocol does indeed meet the criteria and provide at least 95% perfect group membership views under as adverse conditions as 70% packet loss or very high churn rate.**

I. Introduction

Modern vehicles are becoming increasingly more sophisticated with a multitude of sensors, computers and communication capabilities. Advanced driver assistance systems provide improved comfort and increased safety by taking over much of the vehicle control. The first steps to complete autonomous driving are being taken by major automotive companies and road authorities. Inter-vehicle communication (IVC) has the potential to further improve current transportation systems by facilitating efficient coordination between vehicles.

There are numerous potential applications for such technology, including platooning for heavy duty vehicles [4], virtual traffic lights, automated lane merging [19] and highway management, and collision avoidance [20]. However, these applications all require vehicles to form groups and make collective decisions. Since the vehicles are effectively controlled by computers with communication capabilities, this is clearly a distributed systems problem.

Historically much of the research on distributed agreement was motivated by the problem of keeping a consistent state among a set of replicas, thereby providing fault tolerance through replication. The problem of vehicle coordination share some of the characteristics of state replication, also differ in several respects. There are some factors making this problem more challenging such as dynamic interaction groups and very unreliable communication links. On the other hand, satellite navigation protocols provide accurate clock synchronisation to the level of nanoseconds. Moreover, in many cases strict agreement is not required for successful coordination [3], [8]. Often, it suffices for most vehicles to refrain from a certain action for the situation to be safe.

There is, however, a corner stone required for virtually all forms of coordination which is some form of acknowledgement of having received and possibly accepted a request from another node. Provided there is a notion of well-defined groups of collaboration, then having received acknowledgement from all other members in a group allows a vehicle to for example safely perform a lane switch.

In this paper we are concerned with the forming and maintaining of such collaboration groups. Group membership has been extensively investigated as an underlying mechanism for achieving consensus through a concept known as virtual synchrony [7]. However, given the differences between state machine replication and vehicular coordination a new set of requirements are needed also with respect to the performance of group membership.

We propose a new set of correctness and performance criteria for dynamic group membership motivated by the vehicular coordination scenario, but also applicable in other situations where dynamic group formation can occur (including swarms, aerial vehicles, and peer-to-peer systems). The criteria, soundness, completeness and freshness are concerned with properties of views and nodes and intended to be intuitive, well-defined, and verifiable through automated reasoning.

Furthermore, we present a group membership protocol called Synchronous Leader-based Membership Protocol (SLMP) targeted at fulfilling these properties. The protocol has a simple design yet contains some subtleties that make it robust to both packet loss and network churn (nodes leaving and

978-1-5090-5652-1/17/$31.00

joining the system). We assess the performance of this protocol with respect to the proposed criteria through simulation as well as model-based verification. The latter is done with the help of the probabilistic model checker Prism [16]. The results demonstrate that the views provided by the algorithm are always sound. Moreover, the other properties (completeness and freshness) are satisfied in at least 95% of the time unless the system suffers from extreme packet loss (more than 70%).

In summary, we make the following contributions.

- A new set of performance and correctness criteria for dynamic group membership targeted towards collaborative systems.
- A synchronous leader-based protocol for dynamic group membership including an experimental evaluation using simulation.
- Proof of correctness and probabilistic analysis using the probabilistic model checker Prism.

The rest of this paper is organised as follows. Section II presents related work, and is followed by a description of the system model in Section III, where also the proposed view criteria are presented. The proposed SLMP protocol is described in Section IV. Sections V and VI presents the experimental evaluation and model-based verification respectively. Finally, Section VII provides conclusions and future work.

II. Related work

The concept of group membership emerged as an elegant abstraction layer for tackling the consensus problem in distributed systems. While the node crash was the dominating fault model, there has been significant work on tackling network partition faults as well. Transis [2] was the first system that allowed partitions to continue with independent groups. It has been followed by several other such as Totem [18], Moshe [14], Relacs [5], Jgroups [6], Newtop [10], and RMP [13]. For a comprehensive comparison of different group communication services, we recommend the paper by Chockler et al. [9].

Our work is related to leader election since the group membership is dictated by the leader. In traditional leader election algorithms (e.g., see Aguilera et al. [1]), the system model requires a static set of participating nodes, whereas in our work nodes can enter and leave the system dynamically. There are also works that allow dynamic groups with moderate mobility, such as that of Vasudevan et al. [22] but we are not aware of any work on leader election that can handle high churn.

More recently Lim and Conan [17] thoroughly address the problem of group membership in mobile ad-hoc networks. The authors propose a new group membership specification that avoids some of the problems that existed with earlier work. Our work differs from that of Lim and Conan in several respects. First of all, we assume fully synchrony from the onset, which considerably simplify the problem. Moreover, we make no attempt to connect the exact characteristics of which channel properties that will result in group partitions as is done by Lim and Conan. Our approach allows a more performance oriented approach where the ratio of perfect views can be *measured*.

Ferrari et al. [12] also assume synchronous rounds in a cyber-physical system. Their group communication protocol Virtus is built on top of the Low Power Wireless (LWB) bus protocol. This protocol is primarily aimed at very resource constrained devices, with statically encoded nodes, and moderate packet loss. However, while the LWB and SLMP protocols target very disparate types of system, it would not be unreasonable to implement a variant of Virtus on top of SLMP.

There are several other approaches for group membership in vehicular environments including the proposal by Slot and Cahill [21]. They take a hardware-based approach in which laser-scanners provide the vehicle with information about empty areas around it. By combining this information from several vehicles and some non-trivial computation it becomes possible to derive exactly the set of vehicles in a given area.

Fathollahnejad et al. [11] analyse the problem of group formation algorithm in vehicular networks. The authors make a distinction between safe and unsafe disagreement where unsafe disagreement occurs if nodes decide on different non-empty views. They propose a decision algorithm based on very similar assumptions that we make in this paper with synchronous rounds and unreliable communication. However, they also assume the existence of (unreliable) oracles that estimate the number of participating processes. If the oracles never underestimate this number, then the authors show that they nodes will never disagree in an unsafe manner. Similar to our work they use the PRISM model checker [16] to check properties of models where such probabilities are encoded. In their results a packet loss probability of 40% results in approximately 50% chance of agreement on the group and 50% chance of safe disagreement.

Finally, Konur et al [15] present an interesting study where PRISM was used to behaviours of large robotic swarms. Since the behaviour of each robot does not depend on the precise behaviour of the other robots the authors used a counting abstraction to deal with a large number of entities.

III. System model and view criteria

In this section we first establish some basic terminology and system model assumptions, and then proceed to present the three view criteria that we believe are suitable for assessing correctness and performance of dynamic group membership in vehicular networks.

We consider a system to be composed of a dynamic number of vehicles, which we will refer to as *nodes* in line with mobile network terminology. As the next generation of vehicles will by default be equipped with a satellite navigation system, we assume all nodes to be synchronised in time to the necessary precision that allows a synchronous system model to be adopted.

Since the set of nodes is dynamic, we model it as a time dependent function $N(t) = \{n_1, \ldots n_{|N(t)|}\}$, with a restricted rate of change, such that the average rate of nodes entering the

system is λ and each node has a departure rate of λ/N where N is the steady-state number of nodes in the system. The rate of nodes entering and leaving the system is called *churn*. We assume for simplicity that a node never re-enters the system, as this can be modelled by a longer period of message loss.

Nodes can exchange messages that will either be received or dropped. When evaluating the protocol we will assume that a message is received by a node with a fixed probability p. Since we model a dynamic set of nodes we do not explicitly model node crashes, they are simply considered as departing nodes.

We define a *view* as a tuple $v = (l, M, c)$ where l is a leader node, M is a set of members (where $l \in M$), and c is an application-specific topic (or context). A topic can be a particular platoon or the virtual traffic light for some particular crossing. Correspondingly each node i also has a topic denoted $topic(i)$, which we without loss of generality assume to be static, as well as a leader $leader(i, t)$ that can change dynamically.

Note that for the purpose of presentation we have not included monotonically increasing identifiers to the views which is a requirement for some of the basic group membership requirements in the literature. Such information can easily be included, but this is not the focus of this paper.

We are now in a position to define the view criteria. The first (soundness) is a correctness criteria meaning that no view should ever violate this property. The other two should be considered as performance criteria since while meeting them is desirable, mobility, faults and imperfect communication make it impossible to guarantee that they will always hold. A view that satisfy all three criteria is *perfect*.

- **Soundness** A view $v = (l, M, c)$ is sound at time t if all nodes i in M agree on the same topic and leader ($\forall i$ $topic(i) = c$ and $leader(i) = l$)
- **Completeness** A view $v = (l, M, c)$ is complete at time t if for all nodes in $i \in N(t)$ such that $topic(i) = c$ they are also part of the view ($i \in M$).
- **Freshness** A view $v = (l, M, c)$ is fresh at time t if $i \in N(t)$ for every $i \in M$.

Soundness guarantees that all views are disjoint (since a node cannot have two topics, or two leaders) and that all nodes that are in a view are so voluntarily. Completeness states that all nodes that *should be* part of a view (i.e., they have the same topic) are part of the view. A view can become incomplete when a new node enters the system with the same topic as the view before being included in the view. A view can also become incomplete when nodes erroneously believe that another node has left or crashed due to packet loss. Finally, freshness can be seen as the dual of completeness. When a node crashes or leaves the system, it should no longer be included in the view. A view is fresh as long as all nodes in the view are still present in the system and have not crashed.

These criteria are all concerned with properties of views. They do not consider whether nodes have installed a view or not. While this is an interesting aspect, our goal is to separate the specification of views and the dissemination of these views.

IV. Leader-based dynamic group membership

We now proceed to describe the Synchronous Leader-based Membership Protocol (SLMP).

A. Overview

The purpose of SLMP is to maintain dynamic group membership views for mobile nodes. A view consists of exactly one leader and $f > 0$ followers. Every node i has a leader $leader(i)$ that can be either the node itself $leader(i) = i$ or some other node $j \neq i$. A node can be in one of four states: Leading, Joining, Waiting or Following.

Figure 1 shows a basic example of 10 nodes and 3 views. As can be seen in the figure, every view leader has itself as a leader, and all followers in the view also has the view leader as *their* leader. However, there might also be nodes that do not belong in a view, but which still have another leader.

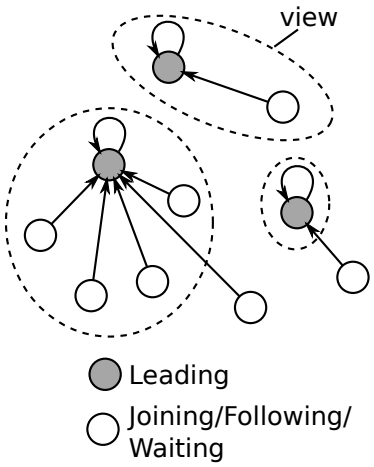

Fig. 1. Illustration of views and leader relations, an arrow from i to j indicates that leader(i)=j

SLMP uses synchronous rounds to send messages to other nodes and to make decisions based on those messages. Figure 2 shows how the time for every node can be divided in three recurring phases, send, act and wait. The send phase is started by the *onSendTimer* interrupt, which should be synchronous over all nodes in the system. The node then proceeds to broadcast the message over the communication channel. Every node that receives the message will invoke the *onReceiveMessage* event. After a predefined send time interval T_s the nodes proceed to the act phase in which nodes change their internal state variables, and then immediately go to the wait phase until a new send phase is initiated.

In SLMP a message m is a tuple $m = (sender, receiver, view, state, leaderAge)$ where sender and receiver are self-explanatory (messages sent to everyone has $receiver = \perp$), the view field sent by a leader is the view (which includes, information about the leader, the followers, and the topic) that will hold in the next round, and the state indicates the current sate of the node. The *leaderAge* field is used by the follower to tell the leader how long it was since it heard from the leader.

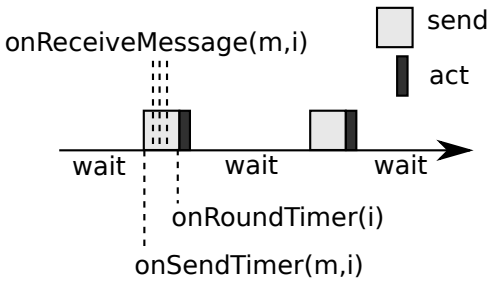

Fig. 2. Synchronous rounds of SLMP

We now proceed to describe the send phase followed by the act phase.

B. Send phase

The send phase is governed by two events, the sending and receiving of messages. Pseudo code of the actions taken by a node upon a send message event is shown in Listing 1. The message sent by a node is determined by the node state (this state variable as well as other variables used in the listsings are global for each node). If the node is in Waiting state, no message is sent. If the node is in the Leading state, a view message is sent with the view that will be valid in the next round (nextView). If the node is in any of the other two states, the node will send out its current view and state, as well as information on how long ago it last recieved a message from the current leader.

The last two rows reset the variables *heardLeader* and *betterLeader* to their initial values in preparation for receiving messages from other nodes (which will result in updating these).

Listing 1 onSendTimer(m,i)

```
1:  procedure ONSENDTIMER(m,i) ▷ message m, vehicle i
2:      if state_i = Waiting then
3:          return
4:      else if state_i = Leading then
5:          m ← (i, ⊥, nextView, Leading)
                ▷ where m=(sender, receiver, view, state, 0)
6:      else                                ▷ Joining or Following
7:          m ← (i, nextLeader, view_i, state_i, leaderAge)
8:      end if
9:      SEND(m)
10:     betterLeader ← false
11:     heardLeader ← false
12: end procedure
```

When a node receives a message m variables will be updated as shown in Listing 2. First, the node checks if the message indicates that there is another leader in the vicinity that is considered better according to some global function *better(i,j)*, which must consider the view topic but can otherwise be an arbitrary total ordering of the nodes (e.g., using node identifiers, or physical location). In our simulations we have used a simple identiy-based method. However it would be interesting to study other variants, including taking the physical location of an entity into account.

If a better leader is found, the *nextLeader* and *betterLeader* variables are set accordingly. Second, if the message came from the node that the node considers as its leader, it first notes that a message was heard from the leader, and also sets the *inView* variables that indicates whether the receiving node was part of the view sent out by the leader or not.

Listing 2 onReceiveMessage(m,i)

```
1:  procedure ONRECEIVEMESSAGE(m, i)
2:      if BETTER(m.sender, nextLeader) then
3:          nextLeader ← m.sender
4:          betterLeader ← true
5:      end if
6:      if m.sender = nextLeader then
7:          heardLeader ← true
8:          inView ← EVAL(i ∈ m.view)
9:      end if
10: end procedure
```

C. Act phase

The act phase is governed by a simple state machine as shown in Figure 3. The state machine consists of the four node states, Leading, Joining, Waiting and Following. All nodes begin in the Leading phase. In the steady state execution of the protocol, there should be one node in the Leading state and the other nodes should be in the Following state.

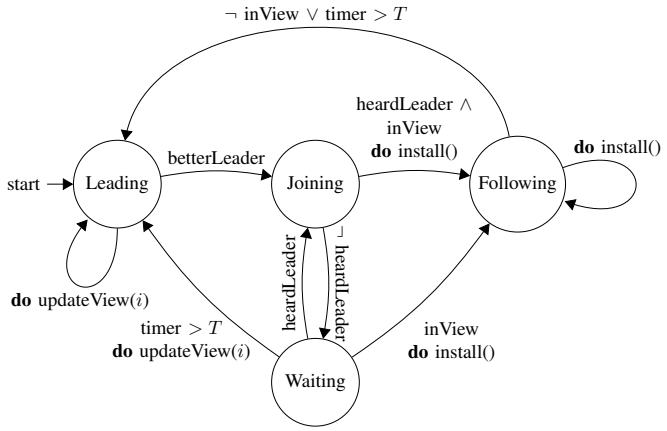

Fig. 3. State machine of the act phase of SLMP

Each time the act phase is initiated the state machine will perform exactly one state transition (potentially resulting in the node staying in the same state).

A node which is in the Leading state can change to Joining state if it learns that there is a better leader (as determined in Listing 2 above). If no better leader has been found, the node performs the *updateView* procedure shown in Listing 3. The purpose of this procedure is to ensure that the view sent out by the leader reflects the set of nodes that have indicated their intent to participate in the view and which have not timed

out. First, the next leader is set to be the node itself, and the next view is initialised with the current view. In lines 4-8, all nodes that have either reached a timeout T are removed from the view. The timer for follower j is called *followTimer[j]* (see section IV-D for details on timers) and the limit T is a tunable parameter that we investigate in the evaluation section. In lines 9-13 all nodes that have declared that they wish to join the group are admitted. In reality such an admission policy could include authentication and authorization, but this is out of scope of this paper. Note that the joining node has already checked that the leader has the correct topic.

Listing 3 updateView(i)

```
1:  procedure UPDATEVIEW(i)
2:      nextLeader ← i
3:      view ← nextView
4:      for all j ∈ view do
5:          if followTimer[j] > T then
6:              nextView ← (nextView \ {j})
7:          end if
8:      end for
9:      for all m ∈ receivedMessages do
10:         if m.state = Joining ∧ m.receiver = i then
11:             nextView ← (nextView ∪ {m.sender})
12:         end if
13:     end for
14: end procedure
```

Proceeding with the state transitions in Figure 3, a node i in the Joining state will stay there until it receives a view $v = (l, M, c)$ from the leader such that $i \in M$, in which case it transitions to the Following state. If no such view is received and the leader is not even heard from, the node transits to a Waiting state in which it prepares to go back to a Leading state. The waiting state can be seen as having the same role as the Joining state, but where no messages are sent out. The reason for having this extra state is to avoid a situation where the node gives up on its intended leader in the same round as the leader finally sends a view where the node is a member.

When a node enters (or stays in) the Following state, it runs the *install* procedure shown in Listing 4. This procedure simply sets the view of the node to the view sent out by the leader.

Listing 4 install

```
1: procedure INSTALL
2:     m ← last received message from nextLeader
3:     view ← m.view
4: end procedure
```

D. Timers

The algorithm contains two timeouts, one where a node gives up waiting for a view from the leader (either from a Waiting state or from a Following state), and one where the leader removes a node from the next view. It is critical for view soundness that these timeouts are properly dealt with. The follower must never abandon the leader before the leader abandons the follower. This is why the *leaderAge* field sent by the followers is needed. When a follower receives a message from the leader it resets its timer to 0. When the leader receives a message from a follower j it sets the *followTimer[j]* to $leaderAge + 1$.

V. Experimental evaluation

In this section we present the results of a simulation-based study on the performance of SLMP. We structure this section in three parts. First we present the experimental setup and methodology. We then show how the protocol is affected by varying group size and dynamics (churn). Finally, we show how the protocol is affected by varying packet loss.

A. Experimental setup

The purpose of this study has been to assess the protocol as a distributed algorithm. Therefore, we have prioritised being able to run many experiments with many rounds rather than high-fidelity packet simulations that would provide details on energy consumption, throughput and protocol overhead. We have developed a custom simulator in C++ that corresponds to the system model outlined in Section III[1]. The simulator maintains a set of active nodes each of which run the SLMP protocol. Messages are exchanged in a round-based manner, and all nodes have access to a global clock. Messages are broadcasted and received with probability p (i.e., a message may be received by only a subset of the nodes). Nodes join and leave the system according to a Poisson process with arrival rate λ and a departure rate λ/N where N is the initial (and also steady-state) number of nodes in the system.

The parameters have been chosen to resemble a highway situation where vehicles that travel in the same direction form groups for collaborative cruise control.

TABLE I
DEFAULT PARAMETERS

Parameter	Value
Initial number of nodes (N)	50
Average group size	5
Communication rate	10 rounds/second
Packet loss probability (p)	40%
Arrival rate (λ)	6 nodes/minute
Simulation length	1000 seconds
SLMP timeout (T)	1 second

Table I show the default parameters used in the experiments. The average group size is controlled by the number of different topics in the system, $G = N/C$ where G is average group size and C is the number of topics. A node randomly chooses a topic upon entering the system and does not change topic.

Each datapoint corresponds to simulation of 1000 seconds (10,000 message rounds) where the first 50 seconds are removed from the results to eliminate boundary effects. In each

[1]The source code for the simulator, the protocol, and the Prism model are available as open source from http://www.ida.liu.se/~mikas34/Software, experiment data is available to the community on request.

round all the views that have been broadcast by some leader are assessed for soundness, freshness and completeness.

B. Impact of group size and dynamics

We know from earlier work [11] that increasing the group size can have drastically negative effect on the ability to agree on a view. Therefore, we study how the view performance is affected by group size as well as by node churn.

Figure 4 shows the impact of varying the average group size (which in turn is decided by the number of topics). The diagram shows four plots corresponding to the desired view properties. As expected, all views are sound. However, the ratio of fresh views decrease as the groups become larger. This is naturally explained by the fact that larger groups are more likely to contain a recently departed node. This explanation holds also for incomplete views which are more likely the more nodes that subscribe to the same topic. The lines for fraction of complete and perfect view almost overlap since there are very few views that are complete but not fresh. Note that the Y-axis of the graph starts at 70% for increased visibility. While these results are very much expected, they confirm that effective vehicle coordination should be done with groups of 10 or less nodes.

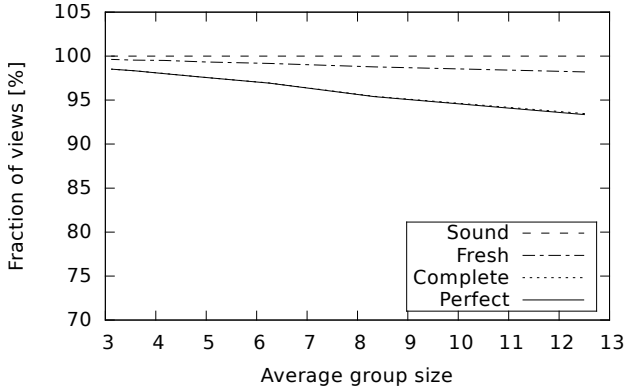

Fig. 4. View performance as a function of average group size

Figure 5 shows the impact of node churn on the SLMP performance. Again the Y-axis starts at 70% and all lines keep above 95% probability. The last point in the curve 18 nodes per minute means that a new node enters the system almost every 3 seconds (with 40% packet loss). Obviously, maintaining perfect group membership in such circumstances is challenging.

C. Impact of packet loss

The other main factor affecting the performance of dynamic group membership is obviously packet loss. SLMP is designed to cope with high packet loss by having a timeout controlling how long to wait before excluding a node from the membership view. Obviously there is a tradeoff between detecting a departed node early and being not spuriously excluding nodes due to lost messages.

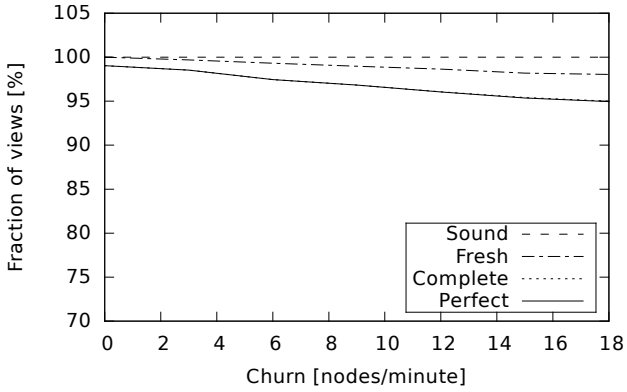

Fig. 5. View performance as a function of churn rate

Figure 6 shows the fraction of perfect views (i.e., views that are sound, complete and fresh) as a function of packet loss probability. The different plots show how SLMP behaves for varying timeout parameter settings. As expected, higher timeout means better resistance to packet loss. The default setting of 1 second timeout which means that a departed node will be excluded after 1 second seems to cope well with packet loss up to 40-50%.

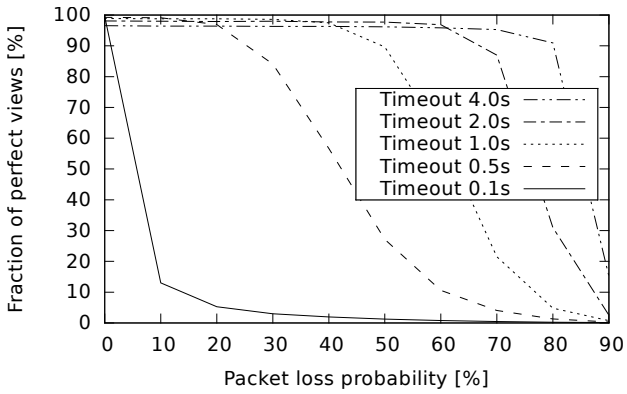

Fig. 6. Ratio of perfect views as a function of packet loss

Increasing the timeout means slower reaction to node changes (in particular departing nodes), which in turn has an effect on view freshness. The optimal timeout setting thus depends on the channel characteristics. Figure 7 shows the ratio of perfect views as a function of the timeout parameter where the different plots represent different packet loss probabilities. For each line, there is an optimum. If the packet loss is 40%, then the optimum lies around 1 second timeout, whereas a 80% packet loss as the highest point at a timeout of 6 seconds.

Since the simulator does include a detailed model of the network stack, we cannot use it to conclude anything about the overhead cost of the protocol. However, there are some insights with regard to the message complexity that is worth mentioning. Each node sends a single message in each round irregardless of the number of nodes. Moroever, only two

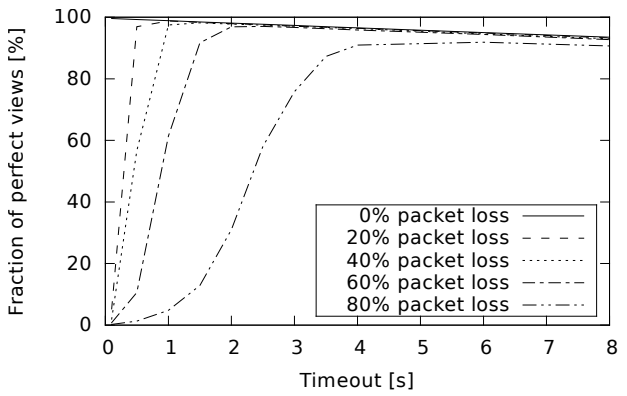

Fig. 7. Ratio of perfect views as a function of the timeout parameter

rounds are needed to form a membership view in case of reliable communication. If messages can be lost, the number of rounds increase, but as is shown in the next section, there is a 95% probability of establishing a view in 10 rounds (1 second) with 20% packet loss.

VI. Model-based verification

In addition to the simulation-based study we also modelled SLMP as a discrete time Markov chain model using the probabilistic model checker Prism (version 4.3 beta). The purpose of this model was twofold. First, this allowed removing critical flaws during the protocol design phase and ensures that the protocol satisfies view soundness. Second, this model allows accurately determining certain guaranteed bounds such as the probability of fast view establishment.

In the rest of this section we first briefly describe the Markov chain model of SLMP, proceed to explain verification of the soundness property, and finally we show some results from the probabilistic analysis of the protocol.

A. Prism model

The Discrete Time Markov Chain (DTMC) model is derived from the state machine shown in Section IV. Multiple nodes are modelled through the module renaming in Prism which allows certain pieces of code to be duplicated. The parallel action of nodes is serialised through a turn variable which dictates for which node the current transition will occur.

Messages exchanges between nodes are modelled through a set of variables. For example, the variable *node_1_2_wants_join* indicates if node 2 has sent a join request to node 1. Similarly, there is a variable *node_1_2_heard* that tells whether node 1 has heard node 2 in this round.

Due to the problem of state space explosion, we have only been able to verify systems with three nodes and at most a timeout of 0.5 seconds (5 rounds). Such a model contains 2,257,449 states and requires over 20 minutes just to be constructed on a computation cluster with 24 2.2GHz cores and 92GB RAM (only one core could be used for model construction).

B. Verifying soundness

The soundness property was verified by asking Prism if the following CTL formula could be satisfied in the initial state.

$$EF \bigvee_{\forall i \neq j} (leader(i) \neq j \wedge inView(i,j)) \quad (1)$$

where $inView(i,j)$ indicates that node i is in the view sent out by leader j. The EF mans that the formula checks whether there exists a path (E) that Finally (F) reaches a state where the rest of the formula holds true. We have verified that for at least 3 nodes this formula is false, meaning that the soundness property cannot be violated (all nodes have the same topic in this model, so that part of the soundness requirement is trivially satisfied).

C. Probabilistic analysis

The DTMC model contains certain non-deterministic transitions, in particular those associated with packet reception or loss. Figure 8 shows the steady state probability of perfect views for varying packet loss probability and four different timeout settings.

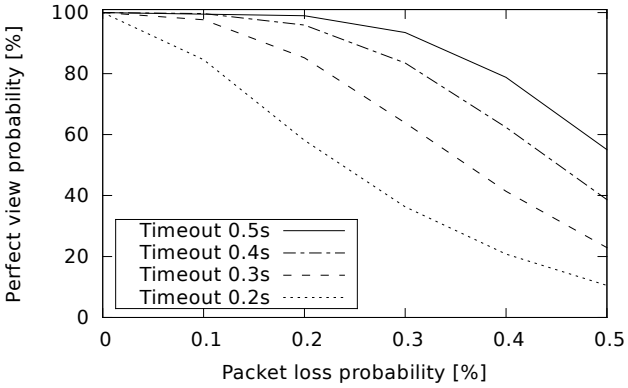

Fig. 8. Steady state probability of perfect views

While this result basically confirms what we could already tell from the simulation-based study, they provide an assured probability which is derived numerically from the model. That is, as long as the packet loss is below 20% the protocol gives a probabilistic guarantee of 99% of producing perfect views.

Figure 9 shows the probability of fast establishment, which we define as being able to form a correct view from an initial state where the nodes have not previously communicated within 250 steps in the DTMC (this corresponds to approximately 1 second in the simulation).

As a final remark, being able to verify the system with only 3 nodes is if of course a limitation. The challenge of automatic formal verification timed distributed algorithms is well known. The state space explosion comes from the fact that each node maintains a state that changes over time (i.e., the timers). However, most protocols currently proposed in the vehicular network field lack any kind of formal basis, so we believe that this is a step forward. Extending the proof to incorporate

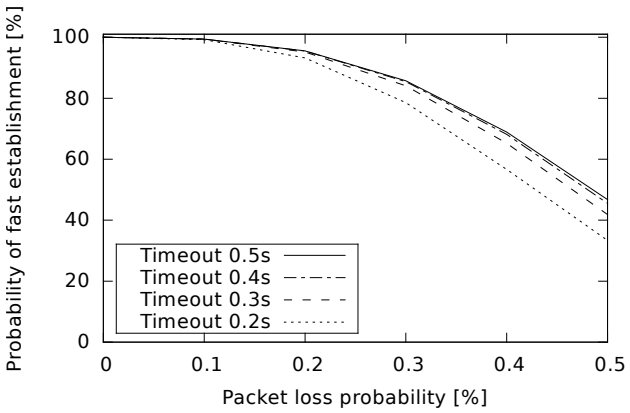

Fig. 9. Probability of fast establishment

an unbounded number of nodes using for example quantifer-based reasoning [3] is one possible direction of future work.

VII. Conclusions

In this paper we have proposed a new set of criteria for dynamic group membership views and a protocol SLMP that fulfil those criteria. The criteria we defined are concerned with the correctness and performance of the views produced by the protocol, soundness, completeness and freshness. When considering whether a set of criteria are useful for specification, there are two important aspects to consider. First, whether it is possible to implement an algorithm or protocol that fulfil the criteria. In this case we have shown (through simulation and model-based verification) that indeed soundness is guaranteed at all times by SLMP, and completeness and freshness in at least 95% of the time even under adverse conditions.

The second important aspect to consider is whether the criteria are strong enough. That is, can a protocol meet the specification, but be worthless in any case? Earlier group membership specification approaches have suffered from this problem by for example allowing singular groups or groups that alternate between singular and useful groups. While we have not proved that perfect views prevent trivial solutions, it seems highly unlikely given that a perfect view must include exactly the set of nodes having the same topic.

In ongoing work we are implementing a reliable group communication layer that runs on top of SLMP. Initial results are encouraging, but more work is required to ensure correctness and bounds on performance. Moreover, it would be of great value to further evaluate SLMP in real vehicular networks and to further develop methods for selecting topics (and to account for dynamic topic changes during the lifetime of a node in the system). We are currently implementing the protocol in a vehicular simulation environment.

Acknowledgment

This work was supported by Centrum för industriell informationsteknologi (CENIIT), project 14.04.

References

[1] M. K. Aguilera, C. Delporte-Gallet, H. Fauconnier, and S. Toueg. *Stable Leader Election*, pages 108–122. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. doi: 10.1007/3-540-45414-4˙8.

[2] Y. Amir, D. Dolev, S. Kramer, and D. Malki. Transis: a communication subsystem for high availability. In *Digest of Papers: Twenty-Second International Symposium on Fault-Tolerant Computing (FTCS)*. #ieee#, 1992. doi: 10.1109/FTCS.1992.243613.

[3] M. Asplund, A. Manzoor, M. Bouroche, S. Clarke, and V. Cahill. A formal approach to autonomous vehicle coordination. In D. Giannakopoulou and D. Mry, editors, *FM 2012: Formal Methods*, volume 7436 of *Lecture Notes in Computer Science*, pages 52–67. Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-32759-9˙8.

[4] AutoNet2030. European fp7 project co-operative systems in support of networked automated driving by 2030. http://www.autonet2030.eu/.

[5] Ö. Babaoğlu, R. Davoli, L.-A. Giachini, and M. Gray Baker. RELACS: A communications infrastructure for constructing reliable applications in large-scale distributed systems. In *Proc. Twenty-Eighth Hawaii International Conference on System Sciences (HICSS)*. #ieee#, 1995. doi: 10.1109/HICSS.1995.375495.

[6] Ö. Babaoğlu, R. Davoli, and A. Montresor. Group communication in partitionable systems: Specification and algorithms. *IEEE Trans. Softw. Eng.*, 27(4), 2001. doi: 10.1109/32.917522.

[7] K. Birman and T. Joseph. Exploiting virtual synchrony in distributed systems. In *Proc. eleventh ACM Symposium on Operating systems principles (SOSP)*. ACM Press, 1987. doi: 10.1145/41457.37515.

[8] M. Bouroche. *Real-Time Coordination of Mobile Autonomous Entities*. PhD thesis, Dept. of Computer Science, Trinity College Dublin, 2007.

[9] G. V. Chockler, I. Keidar, and R. Vitenberg. Group communication specifications: a comprehensive study. *ACM Comput. Surv.*, 33(4), 2001. doi: 10.1145/503112.503113.

[10] P. Ezhilchelvan, R. Macedo, and S. Shrivastava. Newtop: a fault-tolerant group communication protocol. In *Proc. 15th International Conference on Distributed Computing Systems (ICDCS)*. #ieee#, 1995. doi: 10.1109/ICDCS.1995.500032.

[11] N. Fathollahnejad, R. Pathan, and J. Karlsson. On the probability of unsafe disagreement in group formation algorithms for vehicular ad hoc networks. In *Dependable Computing Conference (EDCC), 2015 Eleventh European*, pages 256–267. IEEE, 2015.

[12] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. Virtual synchrony guarantees for cyber-physical systems. In *2013 IEEE 32nd International Symposium on Reliable Distributed Systems*, 2013. doi: 10.1109/SRDS.2013.11.

[13] W. Jia, J. Kaiser, and E. Nett. RMP: fault-tolerant group communication. *IEEE Micro*, 16(2), 1996. doi: 10.1109/40.491463.

[14] I. Keidar, J. Sussman, K. Marzullo, and D. Dolev. Moshe: A group membership service for wans. *ACM Trans. Comput. Syst.*, 20(3), 2002. doi: 10.1145/566340.566341.

[15] S. Konur, C. Dixon, and M. Fisher. Analysing robot swarm behaviour via probabilistic model checking. *Robotics and Autonomous Systems*, 60(2), 2012. doi: 10.1016/j.robot.2011.10.005.

[16] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*. Springer, 2011. doi: 10.1007/978-3-642-22110-1˙47.

[17] L. Lim and D. Conan. Partitionable group membership for mobile ad hoc networks. *Journal of Parallel and Distributed Computing*, 74(8):2708–2721, 2014.

[18] L. Moser, P. Melliar-Smith, D. Agarwal, R. Budhia, C. Lingley-Papadopoulos, and T. Archambault. The Totem system. In *Twenty-Fifth International Symposium on Fault-Tolerant Computing (FTCS-25). Digest of Papers.* #ieee#, 1995. doi: 10.1109/FTCS.1995.466998.

[19] M. Segata, B. Bloessl, S. Joerer, F. Dressler, and R. Lo Cigno. Supporting platooning maneuvers through IVC: An initial protocol analysis for the join maneuver. In *Wireless On-demand Network Systems and Services (WONS)*, 2014. doi: 10.1109/WONS.2014.6814733.

[20] M. L. Sin, M. Bouroche, and V. Cahill. Scheduling of dynamic participants in real-time distributed systems. In *30th IEEE Symposium on Reliable Distributed Systems, SRDS*, 2011. doi: 10.1109/SRDS.2011.37.

[21] M. Slot and V. Cahill. A reliable membership service for vehicular safety applications. In *IEEE Intelligent Vehicles Symposium, IV*, 2011. doi: 10.1109/IVS.2011.5940487.

[22] S. Vasudevan, J. Kurose, and D. Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. In *Network Protocols, 2004. ICNP 2004. Proceedings of the 12th IEEE International Conference on*, 2004. doi: 10.1109/ICNP.2004.1348124.