Membership-based Manoeuvre Negotiation
in Autonomous and Safety-critical Vehicular Systems*
(preliminary report)

António Casimiro [†]   Emelie Ekenstedt [‡]   Elad M. Schiller [§]

June 12, 2019

Abstract

A fault-tolerant negotiation-based intersection crossing protocol is presented. Rigorous analytic proofs are used for demonstrating the correctness and fault-tolerance properties. Experimental results validate the correctness proof via detailed computer simulations and provide a preliminary evaluation of the system performances. The results are compared to the ones that can be achieved via a risk estimator with and without combining the proposed protocol. Our fault model considers packet-loss, noisy sensory information and malicious driving. Our preliminary results show a reduction in the number of dangerous situations and vehicle collisions.

1   Introduction

Autonomous vehicles need a way of assessing risk in order to keep their passengers and other road users safe. Without human inputs, the vehicle has to rely on information from its sensors, and communication with other vehicles and infrastructures. This information must then be interpreted and processed to form a view of the current traffic situation and estimate the likelihood of a future collision, so that the vehicle can take action if the risk gets too high.

Another method of reducing collisions is to let the vehicles follow a protocol that allows them to negotiate their way through traffic situations where lanes either run in parallel or intersect – i.e. where *critical sections* exist. One such protocol for *manoeuvre negotiation* was developed by Casimiro and Schiller [4] and an extended version of this protocol is presented in this thesis. The protocol defines a priority system where vehicles have to request permission from higher priority vehicles to enter a critical section. Granting a request implies giving the requester higher priority. Therefore, the negotiation process provides both a way for the requester to inform other vehicles about its intended manoeuvre and for the requester to ensure that the other vehicles will stop for it if something goes wrong.

Instead of only focusing on collision avoidance, the protocol also tries to prevent *priority violations*. A **priority violation** occurs if a vehicle has to either slow down or stop for another vehicle that is considered to have lower priority according to the traffic rules. The protocol is also designed to keep vehicles safe in the event of network failures, such as when messages are delayed or lost.

---

†Faculdade de Ciências, Universidade de Lisboa, Lisboa 1749-016, Portugal. E-mail: `casim@ciencias.ulisboa.pt`

‡Department of Engineering and Computer Science, Chalmers University of Technology, Gothenburg, SE-412 96, Sweden, `emeeke@student.chalmers.se`

§Department of Engineering and Computer Science, Chalmers University of Technology, Gothenburg, SE-412 96, Sweden, `elad@chalmers.se`.

1.1 Problem formulation

Collision mitigation for autonomous vehicles is an inherently difficult task. The risk assessments needed for mitigating collisions have to be based on information from sensors where noise is always present. The sensors' accuracy may further be reduced by certain weather conditions and objects blocking their view. Allowing vehicles to broadcast their own position to other vehicles may increase the accuracy of the information available for assessing risks. However, for this setup to have a reliable source of information, it is required that no sensor is faulty and that no vehicle has malicious intentions.

The task of collision mitigation can also be performed by making vehicles reveal their intentions instead of trying to infer intentions from positional data. Inferring intentions is a classification problem and it is hard to correctly classify the intention of a vehicle that engages in unusual driving behaviour. Alternately, if vehicle intentions are disclosed via messages, then the system can be exploited by malicious drivers.

To further reduce the number of collisions it is also important that vehicles can negotiate with each other and not only communicate state information. Enabling negotiation implies that the vehicles can collaborate to plan safe trajectories through the sections of road that they will share. Efficiency and fault tolerance can be increased if the negotiating process follows a protocol. To ensure safety, the protocol must be carefully designed and analysed, and undergo rigorous testing. Since the system requires that vehicles follow the protocol to mitigate collisions, this setup is also vulnerable to abuse by malicious drivers.

All approaches to collision mitigation have weak points. Therefore, offsetting these weaknesses by combining different systems with complimentary strengths is critically important if we wish to create a reliable safety system for autonomous vehicles.

1.2 Purpose

As vehicles with autonomous abilities become more and more common, the need for more robust safety systems increases. A lot of research have been made – and still is being made – in the area of predicting vehicle motion and risk estimation. These predictions and estimations can be used to adjust the ego vehicles trajectory in order to avoid collisions. Another approach to implementing a safety system is use communication, vehicle-to-vehicle or vehicle-to-infrastructure, with a protocol determining how messages should be sent. Less time have been spent in this field and in how the two types of safety systems can be combined to create a system less sensitive to noise and network failures etc. The purpose of this thesis is to contribute to the latter field by providing an analytical correctness proof of a safety system based on a communication protocol and by presenting results from running simulations with this safety system, a safety system based on risk estimation, and the two systems combined.

1.3 Delimitations

The implementation of our safety system will only be tested in computer simulations and not on real vehicles. Due to time limitations, the simulations in this project focus only on the highly risky manoeuvre of left turn across the priority lane in an intersection. Another limitation considers the amount of vehicles that are allowed to be in the intersection at the sane time, which is reflected by the number of concurrently granted vehicles. The version of the communication protocol that is presented in this thesis only allows one held grant per vehicle.

1.4 Disposition

A short literary review of various methods for performing risk estimation for autonomous vehicles is found in Chapter 2. Chapter 3 presents the key concepts of the extension of the manoeuvre negotiation algorithm and is followed by a detailed algorithm description and correctness proof in Chapter 4. The software architecture for performing simulations are described in Chapter 6 and the evaluation setup and criteria is presented in Chapter 7. Results from the simulations can be found in Chapter 8 which is followed by a discussion of the

results and suggested further research in Chapter 9. Chapter 10 finally presents the main conclusions of this thesis.

1.5 Contribution

Risk estimation for autonomous vehicles is currently a highly researched topic. A lot of effort has been put into develop individual safety systems for particular traffic situations that could meet the strict regulations on safety for autonomous vehicles. Combining individual safety systems to achieve higher safety has not received much attention yet and it is this area that this thesis will make a contribution to. This thesis presents a way to combine vehicles' individual risk estimation with a distributed, fault tolerant manoeuvre negotiation protocol to increase safety in intersections. The high-level abstraction of the protocol does not limit its usage to intersections and it could potentially be used in many different traffic situations. This thesis also presents an analytic proof of correctness for the protocol. Results from simulations where communication loss and extra noise are added further give an insight into how well the combined safety systems preform in non-ideal situations.

2 Related work

Risk Estimation involves three core components: inferring the behaviour of other vehicles, defining a useful risk metric, and computing the risk based on the inferred behaviour. The aforementioned behaviour may be a composition of the manoeuvre that the vehicle is likely to perform and whether or not it is following traffic rules and conventions.

Detecting high-risk situations in time is vital for avoiding collisions. Another approach to collision mitigation is to design V2V-communication protocols that prevent risky situations from occurring. Here follows a summary of related work in the area of *risk estimation* and *collision mitigation*.

2.1 Behaviour Inference

Machine learning approaches have been shown to function well as classifiers for driver behaviour. A **Recurrent Neural Network** (RNN) with a **Long Short Time Memory** (LSTM) was used by [19] to infer manoeuvre intention at intersections. RNNs have a feedback loop which gives them a form of memory and LSTM cells can remember a value for an arbitrary amount of time, which is useful when predicting manoeuvres.

A RNN classifier was also used to predict driver intention in roundabouts using short segments of LiDAR tracking data [20].

A comparison of different classification methods for *signalled* intersections was performed by [2]. The authors developed two classifiers: one based on a *Support Vector Machine* (SVM) combined with a Bayesian filter and one based on *Hidden Markov Models* (HMMs), having trajectory parameters such as vehicle speed, lateral position etc.

Three classification algorithms based on static time to intersection, the acceleration needed to stop at the stop line, and if the vehicle has a high speed in relation to it's distance from the intersection – compared to a fitted regression curve fit to normal stop-patterns – were also evaluated. The authors found that the SVM performed much better than all the other classification methods, with the HMM in second place. The authors reasoned that the data sets that the HMM used to attempt to generalise driving patterns contained large outliers which strongly affected its results. The SVM, on the other hand, tries to find a *separating boundary* between the sets and thus is less affected by these same outliers.

An **Intention Aware** method was developed by [11] to detect unexpected behaviour in *give-way* intersections. Their method uses a HMM with predefined transition probabilities and a speed model of constant speed. The behaviour of lower priority vehicles is modelled as following or not following the convention of giving way to higher priority vehicles[i].

---

[i]see Section 6.1 for more details

Other aspects of intention aware motion prediction involves the behaviour to follow other vehicles. An intelligent driver model was used by [12] to account for that vehicles usually adapt to the speed of a vehicle closely preceding it.

2.2 Risk Metrics

In this context, the commonly applied definition of risk is **Time to Collision** (TTC), which refers to the time remaining until a collision will occur if no object changes its intended trajectory. The authors of [13] relied on both TTC and another risk metric called **Minimal Safety Margin** (MSM) – which provides a measure of the distance to other vehicles – and managed to detect all risky situations involving lane change on highways in their tests. Their method involved using a Bayesian network to both account for uncertainties and estimate risk for each of the different lanes. They combined this with time-window filtering to produce a robust risk assessment method.

The concept of **Looming** was introduced by [16] to compute risk for *general traffic situations* – i.e. circumstances where predefined trajectories may not be available – that would perform better than mere TTC. Looming measures how much space an object occupies in the *visual field* of a vehicle. A positive loom rate signifies that the object is approaching the vehicle and vice versa. The authors used a **Support Vector Machine** (SVM) trained using a combination of TTC and loom data to detect collisions.

Approaches have been made to not only detect collisions, but also distinguish more severe collisions from minor ones. For this purpose, G. Xi et al [17] presented a situational assessment method that includes the system's internal energy in the risk computation. Internal energy is computed with reference to the vehicle's reduced mass and relative velocity, and provides an indication of the severity of a potential impact.

An interaction-aware model similar to the one in [11] is combined with a **Rapidly-exploring Random Tree** (RRT) classifier by [9] in an attempt to increase safety by paying more attention to the traffic environment. The classifier interprets a traffic situation as either "dangerous" or "safe", enabling a vehicle to be more cautious in situations that are more prone to collisions.

Many Risk Estimation methods are designed for particularly defined traffic situations – such as straight highways or 4-way intersections – but in reality, an autonomous vehicle must be able to handle many different situations. A scenario-adaptive system was developed by Geng et al. [7] to infer driver behaviour of vehicles in more than one traffic situation. A HMM was trained using collected data for a number of differing behaviour classes, with each class corresponding to distinct manoeuvres (e.g. left turn at an intersection, lane keeping etc.). With the introduction of *priori* information (i.e. traffic rules and conventions) and the use of Bayesian inference, [7] also showed that the behaviour prediction time horizon could be increased by on average 56% for lane change and 26% for long time precision. Scenarios were modelled with an ontology model so that scenarios could be inferred using sensory information – to identify features in the environment – and rule-based reasoning.

A model-based algorithm calculating the acceleration or steering angle needed to avoid a collision was presented by [3]. An emergency break or lane-change manoeuvre could be triggered if the output values rise over a set threshold in order to avoid collisions.

2.3 Traffic Coordination Algorithms

Autonomous vehicles can make use of V2V-communication to negotiate and reach an agreement over who was the right to enter critical sections and thereby mitigate collisions. An approach using a **temporarily elected leader** was presented by [15] to direct traffic in intersections. The elected leader decides which direction will have right-of-way, analogous to a green light, until a new leader is elected. The approach introduces a way to control traffic flow through both signalled and unsignalled intersections and was shown to reduce traffic accidents by 70 % in simulations.

Algorithms have also been developed to find an optimal of coordinating traffic through intersections. A **Sequential Quadratic Programming** (SQP) algorithm was developed by [18] for this purpose. The formulated *distributed optimisation* problem considers the times when each vehicle enters and exits the inter-

section, as opposed to considering the complete trajectories, in order to reduce the amount of communicated data between vehicles.

2.4 Cooperative Approaches

Detecting *temporarily occluded objects* is inherently hard since most (often light-based) sensors used in vehicles cannot penetrate opaque materials. Cooperation between vehicles could potentially increase safety, since the object that blocks the sensor's view may actually be another vehicle. Such an approach was developed by [14]. This approach involved vehicles using V2V-communication to inform other vehicles of their own state as well as information about detected – possibly hidden – objects.

3 Solution

This section presents some of the main ideas that have been developed during this project and serves as an introduction to the Manoeuvre Negotiation Protocol presented in Chapter 4. The following sections will cover the main changes made to the original Manoeuvre Negotiation Protocol, introduce the concept of priorities, describe the granting process, and explain how risk estimation is used in our system.

3.1 Adding an explicit release of a received grant

The original protocol for manoeuvre negotiation was designed from a scheduling point of view with focus on both safety and throughput. A permission to change the traffic priorities was bound to a timer set to a predefined manoeuvre time and the permission expired when the timer expired. This approach ensures that no given permission last forever, thus preventing deadlocks, but it assumes that the granted vehicle can finish its manoeuvre within the given time frame. Consider, for example, a granted vehicle p_i that has to stop in an intersection and another vehicle p_j whose grant timer just expired. The protocol would then allow vehicle p_j to be granted access to the intersection, resulting in a collision unless another safety system successfully detects the dangerous situation in time.

3.1.1 From scheduling to setting priorities

Instead of viewing the intersection as a scheduling problem with equal focus on throughput and safety we wish to put more focus on safety by introducing negotiable priorities not bound to timers. The first step towards a priority oriented protocol is to remove the grant timer and to introduce an explicit $\langle RELEASE \rangle$ message to end a grant instead. A granted vehicle will send this release message after it has left the intersection to notify vehicles holding a grant for it to release the grant. This approach assures that a grant will not expire too early and is able to ensure safety even if the granted vehicle slows to a stop in the intersection. Explained from the new priority perspective, a granted vehicle is given a higher priority and none of the granting vehicles, with lower priority, are allowed to enter the intersection until assured that the granted vehicle has left the intersection and they themselves have received a higher priority. For more details on priorities[ii].

The original protocol used timers to prevent deadlocks in the presence of message loss. Removing the grant timer and introducing a new message type implies that an additional feature has to be introduced in order to keep preventing deadlocks. The solution we propose is to let the granting vehicles regularly calculate if the granted vehicle has left the critical section. In this project the calculations are based on positional and road map data sent or shared between vehicles, but it would also be possible to use LiDAR, cameras, and other perception tools for autonomous vehicles for this purpose. A granting vehicle that calculates that the granted vehicle has left the intersection will release the grant in the same way as if it had received a $\langle RELEASE \rangle$ message. A lost $\langle RELEASE \rangle$ message will thereby not cause a deadlock.

[ii]see Section 3.2

3.2 Priorities

The concept of priorities is used by vehicles to determine if they have the right to perform certain manoeuvres when other vehicles are present. More specifically, priorities determine who has the right of way when the planned trajectories of two vehicles intersect.

Priorities can be determined by traffic rules or be indicated by, for example, traffic lights. The former type of priorities do not change over time and we refer to them as *static* priorities. The latter priorities are time dependent and we refer to them as *dynamic* priorities.

Here, we define priorities as the order in which vehicles will be allowed to enter the intersection. The priority order depends on which lane the vehicles are in and how far in space and time they are from the intersection. Our system is based on dynamic priorities where vehicles' priorities are negotiated through our manoeuvre negotiation protocol. Static priorities determined by traffic rules are used as a base state for our dynamic priority system. This section further explains the concept of priorities and how it is used in the Manoeuvre Negotiation Protocol.

3.2.1 Priority matrix

The Manoeuvre Negotiation Protocol relies on manoeuvre specific memberships, which for a vehicle p_i are all the vehicles with higher priority than p_i when p_i wishes to perform the manoeuvre bound to the membership. The default priorities in the negotiation protocol are decided by traffic rules. For an intersection, the traffic rules can be summarised in what we refer to as a *priority matrix*. Each lane leading to the intersection is assigned a priority matrix, where rows correspond to possible manoeuvres and columns represent the other directions into the intersection. Every cell takes on the value True or False. A False implies that a vehicle coming from the direction associated with the matrix and intending to do the manoeuvre described by the row, must ask the agents coming from the lane corresponding to the column for permission to change the traffic priorities temporarily. An example of a priority matrix for vehicles on a straight priority road in a 4-way intersection can be found in Figure 3.1. Vehicles in a lane on the priority road must ask for permission to temporarily change the traffic priorities from vehicles in the opposing lane only if they intend to do a left turn across path, indicated by the False at the top of the first column. The priority matrix for the opposing lane going in the opposite direction will in in this case be the same since the intersection is symmetrical. Vehicles approaching the intersection from a non-priority road will have priority matrix different from the one for vehicles on the priority road.

3.2.2 Priority levels

All vehicles approaching an intersection can be put in a hierarchical order of priority. This order can be visualised as levels on a scale, where the top level determines the group of vehicles that currently have permission to enter the intersection, which are the vehicles with the highest priority. Vehicles in a group of lower priorities will have to wait for higher priority vehicles to exit the intersection before they can enter. This rule is just a direct translation from what the traffic priorities say but the introduction of this concept makes it easier to visualise and perhaps comprehend. The introduction of the Manoeuvre Negotiation Protocol enables vehicles to negotiate with the higher priority vehicles for a temporary rise in its priority and a safe passage through the intersection.

It is worth noting that there can be as many priority levels as there are vehicles on the road since the levels represent an observer's perspective. A vehicle's perspective, however, only needs a maximum of 3 levels corresponding to higher, equal or lower priority than the ego vehicle. This is because a vehicle only needs to compare its own priority to others to navigate through traffic since it just needs to know which vehicles to give way to.

3.2.3 Priority violation

The two main ideas of the Manoeuvre Negotiation Protocol is firstly to let traffic flow in a way similar to when the protocol is not in use when there are no communication failures. Secondly, when communication
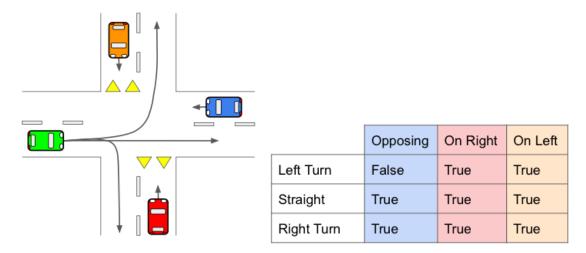
	Opposing	On Right	On Left
Left Turn	False	True	True
Straight	True	True	True
Right Turn	True	True	True

Figure 3.1: A priority matrix for the green vehicle coming in from the left on the priority road. A False infers that the green vehicle has lower priority for the turn described by the row in relation to the lane described by the column.
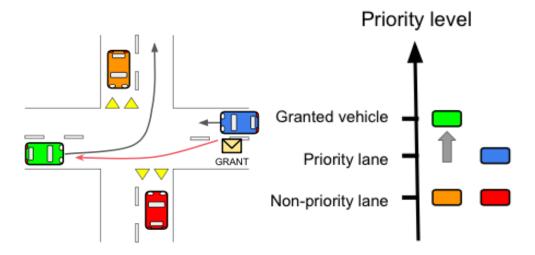

Figure 3.2: The green vehicle coming from the left is granted to temporarily increase its priority level. The green vehicle is now allowed to enter the intersection

failures are present, the protocol is meant to add an extra layer of safety by making the vehicles follow the negotiated dynamic priorities. If, for example, a vehicle has been granted to rise its priority but some circumstance causes the vehicle to stop in the intersection, then other vehicles will not be allowed to enter the intersection until they can confirm that the granted vehicle has left it. This may cause vehicles that initially had higher priority (as specified by traffic rules) to slow down or stop for initially lower priority vehicles. We refer to such events as *priority violations.*

The first main idea can be reformulated to say that priority violations will be avoided when there are no communication failures. To achieve this, our negotiation protocol use the functions noPriorityViolationStrict and noPriorityViolationDynamic to determine if a manoeuvre can be performed without leading to a priority violation. The former takes normal and unusual driving patterns into account while the latter only considers normal driving patterns. More details of the two functions and their usage areas are provided in Section 3.2.4 (Section A for an alternative approach) and 3.2.5.

3.2.4   Algorithm Details of noPriorityViolationStrict

noPriorityViolationStrict is used to calculate which vehicles p_j that have higher priority than p_i – where priorities are defined by the priority matrix for the intersection– and could possibly reach the intersection in the time t_{max} it takes for p_i to both request a priority change and perform its manoeuvre through the intersection. In order to account for unusual driving behaviour, we defined a threshold distance d_{max} such that any vehicle starting from a distance $d > d_{max}$ away from the intersection will not be able to reach it within t_{max} time units.

The implementation of noPriorityViolationStrict(p_i, p_j), for two vehicles p_i and p_j, first determines if p_i has higher priority than p_j according to the priority matrix and if so returns True. If p_i has lower priority the procedure returns True if and only if p_j is further than d_{max} length units away from the intersection.

3.2.5   Algorithm details of noPriorityViolationDynamic

The implementation approach of noPriorityViolationDynamic makes an assumption about driver behaviour in order to try to increase throughput. This approach requires predefined trajectories and speed models corresponding to the manoeuvres going straight, turning left or right, and stopping.

Let p_i and p_j be two vehicles approaching an intersection. A call to noPriorityViolationDynamic$(p_i, turn_i, p_j)$ will return True if the vehicles are estimated to never be in the intersection at the same time. The estimation of time period spent in the intersection – here called the *occupation interval* – is performed for both vehicles but the process is here described for p_j only to avoid duplication. Normal driving pattern for p_j's current intended manoeuvre is used to estimate the time it will take p_j to reach the intersection (TTI_j) and the time it will take for it to leave the intersection (TTE_j). The **occupancy interval** for p_j is then defined as $O_j = [TTI_j, TTE_j]$. The intention of p_j may be unknown, in which case O_j is a set consisting of the predicted intervals for each of the four driving manoeuvres. An additional margin of $\chi(TTI_j - T)$ – where T is the *current time* and χ is a constant – is subtracted from each TTI_j and a margin of $\chi(T - TTE_j)$ is added to each TTE_j to help avoid near collisions. The constant χ affects the uncertainty in the estimation of O_j. Thus, more uncertainty is added to O_j for an agent p_j that is further away in time from the intersection than an agent that is closer. This increase in uncertainty accounts for some deviations from the defined normal driving pattern. We define $\mathcal{O}_j$ as the **increased occupancy interval** for an agent p_j.

In the next step, the increased occupancy intervals for p_i and p_j, $\mathcal{O}_i$ and $\mathcal{O}_j$, are compared to evaluate the traffic situation. No priority violation is estimated to occur between p_i and p_j if $\mathcal{O}_i \cap \mathcal{O}_j = \emptyset$.

This approach cannot guarantee that no priority violation – and thereby that no collisions – will occur if vehicles exhibit behaviour that deviates extremely from the normal driving patterns and priorities are kept static. However, enabling priorities to be dynamic by introducing negotiation according to the *Manoeuvre Negotiation protocol* ensures that no collision will occur as long as all vehicles follow the protocol. This implementation of noPriorityViolationDynamic is therefore suitable for use during the negotiation process

when agents make decisions on granting or denying another agent. The approach provides higher throughput than the approaches presented in Section 3.2.4 and A.

Algorithm 1: noPriorityViolationDynamic relying on dynamic priorities for safety

1 **Structures:**
2 $AgentState = (ta, x, v, a);$ `// timestamp, position, velocity, acceleration`
3
4 **Interfaces:**
5 $getTimeToIntersection(p[, turn]): ;$
6 $getTimeToExit(p, TTI[, turn]): ;$
7
8 **procedure** $getOccupationInterval(p, turn = None)$ **begin**
9   **let** $TTI = getTimeToIntersection(p, turn);$
10   **let** $TTE = getTimeToLeave(p, TTI, turn);$
11   **let** $TTI = TTI - (TTI - p.ts)\chi;$
12   **let** $TTE = TTE + (TTE - p.ts)\chi;$
13   **return** $[TTI, TTE]$
14 **procedure** noPriorityViolationDynamic$(p_i, turn_i, p_j)$ **begin**
15   **let** $TTI_i, TTE_i = getOccupationInterval(p_i, turn_i);$
16   **let** $TTI_j, TTE_j = getOccupationInterval(p_j);$
17   **if** $((TTI_i \geq TTI_j) \wedge (TTI_i \leq TTE_j)) \vee ((TTI_j \geq TTI_i) \wedge (TTI_j \leq TTE_i))$ **then return** True ;
18   **else return** False;
19   **return** $unsafeAgents$;

3.3 Deciding to grant or deny

In our system the traffic priorities at any point in time are determined by the static traffic rules and the dynamic changes of priorities established by the manoeuvre negotiation protocol. A vehicle that wants to perform a manoeuvre that would change the current traffic priorities will therefore have to ask for permission to do so from all vehicles in its membership. The choice of reply to a request, either $\langle GRANT \rangle$ or $\langle DENY \rangle$, depends on the output of the function noPriorityViolationDynamic which computes *occupation intervals* – time intervals in which the requester and requestee are predicted to be inside the intersection.

3.3.1 Predicting intersection occupation intervals

The prediction of our occupancy intervals used in noPriorityViolationDynamic are based on normal driving patterns defined as a speed model, see Figure 3.3. The speed model defines the ideal speed based on a vehicle's intention and its distance from the intersection. Starting from the most recently known position at time T of a vehicle p_i, the time to reach the intersection, TTI_i, and time to exit the intersection, TTE_i, is computed using the speed v determined by the speed model and by integrating $1/v$ from the current position up until reaching the intersection. The uncertainty constant χ, which reflects how well we believe that vehicles follow the speed model, was set to 0.25 here since the vehicles in our simulation do not deviate much from the speed model.

3.3.2 noPriorityViolationDynamic's role in the grant decision

A Requestee calls the interface noPriorityViolationDynamic presented in Algorithm 1 to determine if it's safe to grant the requester. The interface calculates the occupation intervals $\mathcal{O}_i$ and $\mathcal{O}_j$ for the requestee p_i and the requester p_j and calculates the gap Δt_{gap} between them when put on the same time line, see Figure 3.4. Any overlap in the intervals, $\Delta t_{gap} < 0$, signifies a higher collision risk and will result in a returned False causing p_i to reply with a $\langle DENY \rangle$. No overlap among the two time intervals results in a returned True which enables p_i to continue checking the other state related criteria needed when making the grant
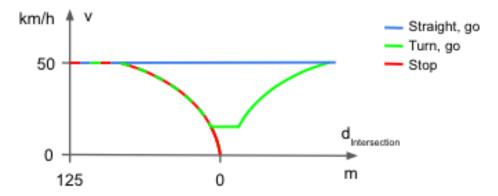
Figure 3.3: A speed model defining ideal speed v for a vehicle based on its distance from the intersection $d_{Intersection}$ and intention which can be either going straight, turning left or stopping. The model is based on alternating constant speed and constant acceleration.
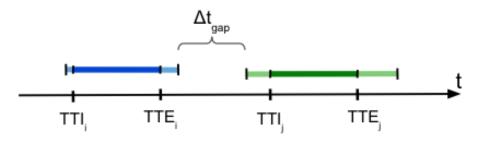
.

decision[iii].

Figure 3.4: Two occupancy intervals for agent p_i and agent p_j with added safety margins. The time gap Δt_{gap} is evaluated as the smallest gap between the two intervals outer margins.

3.4 Risk estimation interfaces

The manoeuvre negotiation protocol needs risk estimation in order to provide safety, especially when deviations and failures are present in the system. Three main usage areas for risk estimation will be presented in this section where the risk estimation can be provided to the manoeuvre negotiation protocol in the form of interfaces.

3.4.1 Decide when to grant or deny a request

The receiver of a priority change request has to decide whether to reply with a $\langle GRANT \rangle$ or a $\langle DENY \rangle$. The decision has to be based on how much risk is involved with the proposed change of priority. In this case we define high risk as an overlap in the estimated time intervals of when the requester and requestee will be in the intersection[iv].

[iii]see Section 5.1.2

[iv]see Section 3.3

3.4.2   Decide when enough responses have been received

The negotiation process relies on memberships to ensure that all agents necessary for ensuring safety are involved in a negotiation process. The memberships are dynamic and are recalculated every T_M time units. A membership retrieved just before sending a request may therefore change within the $2T_D$ time units that the requesting vehicle waits for replies. Some agents may be added and some removed from the membership. An agent that is removed during the waiting time may cause the requesting vehicle to wait unnecessarily, since an agent removed from the membership does not have higher priority anymore and their (possibly lost) reply is no longer relevant. A requesting agent should therefore filter the received replies to check if it has got replies from the intersection of the set of agents in the membership valid when the request was sent $\mathcal{D}$ and the current membership MS to avoid this waiting period. More formally, the vehicle needs to check if $\mathcal{D} \cap M \cap \mathcal{R} = \phi$, where $\mathcal{R}$ is $\mathcal{D}$ minus the agents that have responded to the latest request and ϕ is the empty set.

3.4.3   Risk estimation as a background process

In the real world, outside simulations, one cannot assume that all vehicles will follow an ideal trajectory and follow the traffic rules at all times. An autonomous vehicle thus need to be able to detect a deviating and possible malicious vehicle and perform needed action in time to avoid a collision. Continuously running a risk estimation procedure, that can perform long-term predictions, as a background process increases the chances of detecting these deviating vehicles in time to avoid a collision. For this thesis we are using a slightly modified version of the risk estimation procedure presented in [11][v]. We refer to this process as the *Risk Estimator*. When the Risk Estimator detects a risk over 75% then it triggers an *emergency break*[vi].

3.4.4   Interconnected risk estimator and manoeuvre negotiation

The change of priorities achieved with the manoeuvre negotiation protocol must be recognised by the risk estimator since the risk estimator uses priorities to calculate risk. For example, a vehicle p_i on the priority road that have granted another vehicle p_j should not detect a high risk when p_j enters the intersection before p_i since p_i has agreed to give p_j higher priority. If the risk estimator would not be made aware of the priority change then, depending on the time gap, it might detect a high risk since p_j would be seen as a low-priority vehicle doing a turn in front of a high-priority vehicle.
The protocol calls the interface ntfyEstimat defined in Figure 5.1 with grant to tell the risk estimator about the priority change. The risk estimator puts the granted vehicle in a grant list identifying vehicles with higher priority, until the interface ntfyEstimat is called again from the protocol with revoke.

4   The Manoeuvre Negotiation Algorithm

4.1   System Settings

Traffic rules regulate entry allowances to avoid ambiguity and reduce collision risk when two or more vehicles arrive at an intersection's border. Each vehicle has to arrive at the intersection border in a way that allows the driver to observe all other road users that they must give way to. It is allowed to enter and cross the intersection before higher priority vehicles as long as it does not *violate their priority* – i.e. either cause a collision or force the higher priority vehicles to change speed, path or trigger their safety systems to perceive risk.
We consider a road intersection with *predefined priorities*. These priorities are determined by the *give way* traffic rules, such that whenever two vehicles have intersecting paths, one of them has a predefined (strictly) higher priority that depends on the incoming lanes of the two vehicles.

---

[v]see Section 6.3.3
[vi]see Section 6.3.3 for its definition of risk

An agent is a *computing and communicating entity* that assists in deciding whether a vehicle may cross the intersection. Such algorithmic agents can be components in a distributed automated driving system. Note that our model assumes that all vehicles are controlled by these agents and it does not focus on how to deal with malicious agents in a way that can represent (careless) human drivers, since we assume that all agents are attentive and follow the traffic rules. We also assume that the agents periodically transmit information about their whereabouts over communication channels that are prone to failures.

Agents can perform manoeuvres concurrently, such that each agent performs a sequence of manoeuvres; one manoeuvre at a time. Agents have autonomous abilities that enable them to keep to their lanes and to not drive too close to the vehicles in front of them in the same lane. We assume that the agents have collision avoidance mechanisms. However, these mechanisms cannot facilitate (comfortable) intersection crossings since these mechanisms do not assess the risk of entering the intersection in a way that violates the traffic rules. For example, each agent must give way to higher priority traffic. This work focuses on agents' obligations to stop, if necessary, and let higher priority agents proceed. The exact decision as to whether a complete stop is needed depends on the traffic rules and situation. This can be determined by a separate algorithmic (non-distributed) component for risk estimation, such as the one by Lefèvre et al. [11], which in turn depends on manoeuvre prediction and other components. Our model assumes access to such a risk estimator and that the vehicle changes its speed (but not its path) whenever risk is perceived.

Definition 4.1 *Suppose that agent p_i must give way to agent p_j since p_i has lower priority than p_j. We say that a* priority violation *has occurred if, and only if, p_i decides to enter and cross the intersection before p_j in a way that causes p_j's risk estimator to observe the need to reduce speed.*

We note that in the presence of aberrant behaviour – such as that caused by vehicular malfunction or malicious intent – it is impossible to guarantee that no priority violation (Definition 4.1) will occur. For example, a vehicle may enter an empty intersection without any approaching vehicle in sight. Any aberrant event may then cause the vehicle to come to a permanent halt within the intersection, thereby blocking the way for other vehicles. For the sake of making the scope manageable, Assumption 4.1 states that such events do not occur. Our model assumes that agent p_k has access to the function noPriorityViolationDynamic(@*parameters*), where @*parameters* refers to timed information about the computational context, such that Assumption 4.1 is respected. The *time aspects* defines a time horizon and concerns the time it will take a requesting agent to send and receive answers to a request and how long it will take to perform the requested manoeuvre. The function noPriorityViolationDynamic must then evaluate the situation up until this time horizon.

We have considered two different implementations of noPriorityViolationDynamic that provide different accuracy to throughput ratios. The first solution is based on patterns for normal driving in approaching an intersection. Speed models are used to estimate when vehicles will enter and exit the intersection. Vehicles deviating from the normal driving pattern are accounted for by adding an extra safety margin based on how far into the future the entry and exit occurs. Overlaps in the estimated intervals implies a potential priority violation. This approach is elaborated in Section 3.2.5.

The second approach is based on occupancy prediction [1] and should be able to identify more situations in which priority violations can occur. However, the accuracy of predicting a future priority violation is favoured over keeping the throughput through the intersection high. Further details are given in Section A.

Assumption 4.1 *Suppose that the* requester *agent p_i requests a permission at a time t_i from all relevant* requestee *agents that have a higher priority. Moreover, p_j receives p_i's request at time t_j, asserts that* noPriorityViolationDynamic($AR, turn$) = True *and then replies to p_i with a $\langle GRANT \rangle$, where t_j is between t_i and $t_i + T_D$, and T_D is a bound on the communication delay of a message sent from one agent to another. Furthermore, by the time t_i', agent p_i has received $\langle GRANT \rangle$ replies from all the relevant agents, where t_i' is a time between t_i and $t_i + 2T_D$. In this case, we assume that the fact that agent p_i starts crossing the intersection immediately after time t_i', implies that it is safe to cross and no priority violation will occur.*

We note that some benign violations of Assumption 4.1 are not deemed to violate system safety as a whole. Specifically, consider the case where agent p_i, before time t_i', stays outside the intersection and immediately after time t_i' starts crossing the intersection and halts (unexpectedly and without a notification) before

completing the manoeuvre. The use of a risk estimator, such as the one by Lefèvre et al. [11], allows ample time for the arriving agent p_j to safety avoid a crash with p_i. To the end of accommodating the ability of avoiding such benign violations of Assumption 4.1, we assume that the algorithm can notify its (local) risk estimator module (by calling ntfyEstimat($state$) : $state \in \{\mathsf{grant}, \mathsf{revoke}\}$) about temporary changes to the priority setting. E.g., if p_j grants p_i a permission to cross the intersection before itself then p_j calls ntfyEstimat(grant), and once p_j learns about p_i's exit from the intersection then p_j calls ntfyEstimat(revoke). In detail, Lefèvre et al. [11] refer to behaviour that is expected from the agent p_j when it has a priority over p_i. We simply assume that this priority information is an input to the risk estimator module that can change during the period in which p_i is granted a permission to cross the intersection before p_j. We note that without this assumption, a risk can be falsely detected if p_i starts to cross the intersection before p_j.

4.1.1 Synchrony in the presence of communication failures

All agents have access to a universal clock. We assume that the system uses mechanisms for omitting messages that do not arrive in a timely manner. That is, a timely message is delivered at most T_D time units after being sent. Non-timely messages are not processed by the algorithm. Let E be a system execution in which any sent message is delivered in a timely manner, i.e., no message is lost or omitted, and all delivered messages are sent during E. In this case, we say that E is a *synchronous* execution. All other executions are simply referred to as *non-synchronous*. We assume that during non-synchronous executions, the *communication is fair*. That is, any message that is sent infinitely often arrives infinitely often in a timely manner. However, we do not bound the time between two consecutive timed message arrivals. We note that the latter implies unbounded, yet finite, message delays. Since we consider the general context of wireless communication, we assume that the system without notice can reach a non-synchronous execution for any unbounded (yet finite) period.

4.1.2 Task requirements

The correctness proof will demonstrate both safety (Requirements 4.1, 4.2, 4.3, 4.4, and 4.5) and progression (Requirement 4.6) for synchronous executions. For the case of non-synchronous executions, the proof considers the demonstration of safety (without considering progression).

4.1.3 Limitations

For the sake of simple presentation, we make the following assumptions. We assume that an agent can request only one manoeuvre at a time since we consider the task of multiple manoeuvre requests to be related to planning. We consider a system that at any time allows at most one grant to a single agent; systems that consider concurrent grants are relevant extensions to the proposed solution. Once an agent is granted a permission to perform a manoeuvre, we require the system to allow the agent to either (i) finish this manoeuvre within T_{Man} time units or (ii) to communicate its position in a timely manner to all agents that might collide with it (and by that notify them about the delay in the termination of the manoeuvre). We note that a possible extension to the proposed solution exists in which the system can suspect any granted agent that does not fulfil (i) and (ii) to be faulty. Such extensions can, for example, trigger a recovery procedure that makes sure that the road (intersection) is clear before revoking the grant and resuming normal operation.

4.1.4 Cloud-based membership service

Requirement 4.1 considers a Cloud-based service that computes new memberships every T_M time units and performs the computation on state data sent periodically from all agents to a storage service. A *membership* for an agent p_i contains a set of identifiers of agents p_j whose: (i) path can interest with the one of p_i's paths, and whose (ii) lane has equal or higher priority than p_i's lane, and whose (iii) distance to p_i is at most d_{max} length units. Since the priorities are determined by direction specific traffic rules, we assume that the membership information provides one membership per possible manoeuvre through the intersection.

The Cloud-based service associates a timestamp ts with every instance M of a membership. The timestamp of the oldest agent state data of the agents in M determines ts. Moreover, we say that M is *fresh* if $T < ts + 2T_M$, where T is the current time. When agent p_j appears in p_i's fresh membership M, we say that p_j may cause a *priority violation* with p_i.

Every membership M of an agent p_i is also associated with a boolean value MO, which we call the *Manoeuvre Opportunity* indicator. The membership service sets $MO = \mathsf{True}$ to indicate that all agents in the membership set are within the maximum communication range of p_i or that the provided membership for some other reason should not be considered valid. Note that the latter assumption allows the Cloud-based service, when needed, to indicate that the functionality offered by the proposed solutions is temporarily unavailable due to some implementation constraints. A concrete implementation of the proposed solution might, for example, require the assumption of an upper bound, N_{max}, on the number of vehicles that can occupy the considered traffic scenario due to limitations related to space, and communication range. In addition to violations of N_{max}, other constraints could consider the occurrence of failures or limitations on velocities in which the offered functionality cannot be considered feasible and safe.

Requirement 4.1 (Membership Service) *Let M be a membership for an agent p_i that has a manoeuvre opportunity at a time T and M is fresh: $T < ts + 2T_M$. We require that M contains all agents that could potentially lead to a priority violation with p_i from T to $T + 2T_D + T_{Man}$.*

4.1.5 Requester-to-requestee dialogue

We consider a pair of distributed state-machines that the requester and requestees execute; cf. agents p_i, and respectively, p_j in Figure 4.1. We require that these agents use noPriorityViolationDynamic() in a way that allows using Assumption 4.1 safely and repeatedly. To that end, there is a need to take into consideration both $\langle DENY \rangle$ messages and temporary timing failures due to the occurrence of non-synchronous executions. Therefore, the requester automaton includes a pair of states for facilitating a persistent transmission of request messages $\langle GET \rangle$. Thus, whenever it is time to send a request, the requester enters the state GET directly or passes through state $TRYGET$ first before reaching state GET. From state GET p_i repeatedly multicast to all relevant requestees the message $\langle GET \rangle$. Moreover, once requester p_i recognise that it has received a $\langle DENY \rangle$ message, p_i defers communication by entering the state $TRYGET$ and stays there until it is time to return to GET. We note that all agents play both roles of requesters and requestees; the proposed solution does not distinguish between these roles. For the sake of simple presentation, the requirements presented here mostly follow the above role separation with one exception; Requirement 4.3 refers explicitly to the case in which a requester receives a request and thereby also has to play the role of a requestee.

Requirement 4.2 (Request Initialisation) *Consider an agent p_i that aims to perform a manoeuvre through an intersection and a membership M that corresponds to the direction of that manoeuvre. We require p_i to first request a permission from all agents in M according to the required dialog presented in Assumption 4.1. Moreover, we allow p_i to initiate such a request if (i) M is fresh, and (ii) $MO = \mathsf{True}$. Otherwise, we require p_i to check conditions (i) and (ii) within T_A time units.*

Requirement 4.3 uses the term *status* when referring to the automaton state and assumes that the state of an agent can encode the identifier of the agent to which it has granted a permission. It also assumes the existence the function myPriority(m) that can arbitrate between two concurrent requests. This function returns True if the current request of the calling agent has higher priority than the request encoded by message m. The implementation of myPriority() can depend on the directions of the relevant agents, the time in which their requests were initiated as well as the agent identifiers.

Requirement 4.3 (Granting and Denying) *Suppose that the requester agent p_i multicasts a permission request, message $m = \langle GET \rangle$, at a time t_i, to all requestee agents in $\mathcal{D} = M$, where M is p_i's fresh membership at the time t_i for which there is a positive indication for a manoeuvre opportunity. Moreover, p_j receives p_i's request at a time t_j, where t_j is between t_i and $t_i + T_D$. We require agent p_j to call the function*
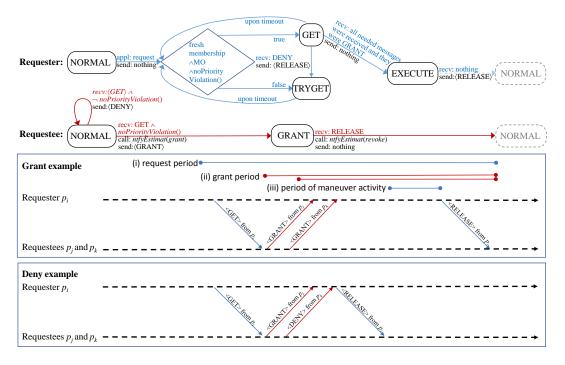
Figure 4.1: The dialog between the requester and the requestee depicted by the state-machines above and below respectively. Examples for grant and deny processes are described in the message sequence charts provided above and below respectively. The grant example also shows the periods in which (i) the request is ongoing, (ii) the grant has changed the state of the requestees so that the requester has the priory over the requestees, and (iii) when the manoeuvre is performed.

ntfyEstimat(grant) *and reply with a* $\langle GRANT \rangle$ *to* p_i *if, and only if,* noPriorityViolationDynamic() $=$ True *and:* *(i)* p_j*'s status is* $NORMAL$ *or* $TRYGET$, *(ii)* p_j*'s status is* $GRANT$ *and its state encodes that the grant is already given to* p_i, *or (iii)* p_j*'s status is* GET (p_i *and* p_j *have concurrent requests) and* p_i *has the higher priory according to* myPriority(m). *Otherwise,* p_j *replies to* p_i *with a* $\langle DENY \rangle$ *message.*

Requirement 4.4 (Releasing) *Let* p_i, p_j, $\mathcal{D}$, *and* M *be defined as in Requirement 4.3. We require agent* p_i *to multicast a release message,* $\langle RELEASE \rangle$, *to all requestee agents* p_j *in* $\mathcal{D}$ *only in the following cases:* *(i)* p_i*'s status is* GET *and it receives a message* $\langle DENY \rangle$ *from* $p_j \in \mathcal{D} \cap M$, *(ii)* p_i*'s status is* GET *and its timer expires so that it needs to defer its requests by changing status to* $TRYGET$, *(iii)* p_i*'s status is* $EXECUTE$ *and it has just left the intersection, and (iv) case (iii) of Requirement 4.3 occurs. We also require agent* p_j *to call* ntfyEstimat(revoke) *only in two special occasions. The first occasion is when* p_j *status is* $GRANT$, *its state encodes that it has granted permission to agent* p_i *and it has just received the message* $\langle RELEASE \rangle$ *from agent* p_i. *The second occasion occurs when* p_j*'s status is* $EXECUTE$ *and* p_j *has left the critical section.*

Consider an agent p_i that has requested a permission from all agents in $\mathcal{D}$ (when the value of p_i's membership set was $\mathcal{D}$, cf. Requirement 4.2). Suppose that G is the set of agents whose responses arrived to p_i from $p_j \in \mathcal{D}$ within $2T_D - T_A$ time units, and M is p_i's current membership. We say that p_i is *fully-granted* when (i) $G \supseteq \mathcal{D} \cap M$ and (ii) all the agents in G responded with a $\langle GRANT \rangle$ or if at least $2T_D$ time units have passed since the request was first sent (to the agents in $\mathcal{D}$). Whenever G includes an agent that responded with a $\langle DENY \rangle$, we say that p_i is *fully-denied*. In all other cases, we say that p_i's request is *pending*.

Requirement 4.5 (Allowing and Deferring Manoeuvres) *We require the agent p_i to: (i) only perform a manoeuvre that is fully-granted and to send a release message upon its completion, and (ii) in the case of a fully-denied manoeuvre, send a release message to all agents in $\mathcal{D}$ and restart the request process after T_A time units.*

Suppose that there is a (possibly unknown, yet finite) bound, $T_{admissible}$, such that a period of $T_{admissible}$ time units in which p_i's membership is fresh and empty implies (i) a safe crossing opportunity for p_i during that period (in the sense that p_i's membership is empty, the manoeuvre opportunity indicator is True and it has not granted any request for a period that overlaps with this period of $T_{admissible}$ time units) and that (ii) agent p_i has the time to perform the manoeuvre (in the sense that p_i has the time to notice that its membership is fresh and empty, and the manoeuvre opportunity indicator is True). We call such a crossing opportunity an *admissible crossing opportunity*. We note that the fulfilment of Requirement 4.1 implies the existence of $T_{admissible}$.

Requirement 4.6 (Progression) *Let E be a synchronous system execution in which admissible crossing opportunities occur infinitely often (for every direction of the intersection). Moreover, suppose that, infinitely often, the agents that are the closest to the intersection, aim at crossing the intersection. Then, a fully-granted agent crosses the intersection infinitely often.*

The constants considered in the proposed algorithms and that are mentioned in the Theorems and Lemmas are provided in Table 5.1.

5   Algorithms

The time constants used in the proposed algorithms are provided in Table 5.1. The following internal size order is implied: $T_{Man} > T_D \geq T_A$ and $T_M > T_A$.

T_M	Period of membership protocol
T_A	Period of agent registry update
T_D	Upper bound on transmission delay between vehicles
T_{Man}	Upper bound on manoeuvre execution time

Table 5.1: Time constants used in Algorithm 2 and 3. Observe that we assume that $T_{Man} > T_D \geq T_A$ and $T_M > T_A$.

5.1   Algorithm description

This section provides a line by line description of the Membership Algorithm (Algorithm 2) and Manoeuvre Negotiation Algorithm (Algorithm 3).

5.1.1   Description of the Membership Algorithm

The **Membership Algorithm** (Algorithm 2) periodically updates the membership set for every vehicle in a specified geographical section called a *segment*. A **membership set** holds multiple memberships. A membership in the membership set for a vehicle p_i consists of a set called Safety Membership ($\mathcal{SM}$) made up of the IDs of all other vehicles of higher priority in the same segment of the road, a flag called Manoeuvre Opportunity (MO) that indicates whether or not the membership is valid, and a time stamp. There has to be one membership per manoeuvre p_i can perform in the segment since traffic priorities are manoeuvre-based and we want the protocol to preserve the initial traffic priorities if possible. The segment in this case is a 4-way intersection. The possible manoeuvres therefore correspond to turning left, going straight, and turning right.

Algorithm 2: The membership algorithm.

1 Structures:
2 $Segment = (xs, xe)$;   `// positions: start, end`
3 $AgentState = (ta, x, v, a)$;   `// timestamp, position, velocity, acceleration`
4 $Agent = (\mathsf{aID}, AS)$;   `// agent ID, AgentState structure`
5 $Membership = (tm, MO, \mathcal{SM})$;   `// timestamp, MO flag, `$\mathcal{SM}$` IDs of agents in the membership`

6 Constants:
7 sID: the segment ID;
8 $S := \mathsf{getSegmentData}(\mathsf{sID})$;   `// Segment registry`

9 Variables:
10 $\mathcal{MR} \leftarrow \emptyset$;   `// Membership registry`

11

12 Interfaces:
13 clock(): read current time;
14 getSegmentData(sID): read Segment registry for segment sID from the storage service;
15 getARSet(sID): read all Agent Registries for segment sID from the storage service;
16 storeMR(aID, MR): write MR (Membership registry) for agent aID on the storage service;
17 getHigherPriorityAgents(AR, t): uses noPriorityViolationStrict (Section 3.2.4) to obtain the sets of agents with whom the agent aID identified by AR may cause a priority violation for the different possible turns defined by getPossibleTurns(AR). We assume that this set includes all the agents of higher priority than aID that could possibly arrive to the intersection by the time t;
18 getReachableAgents(AR, t): obtain the set of agents that will be within communication range from the agent identified by AR until t. ;
19 getPossibleTurns(AR): obtain a list of possible turns for an agent;

20

21 do forever (once in every T_M time units) begin
22   let $\mathcal{A} = \mathsf{getARSet}(\mathsf{sID}())$;
23   **foreach** AR *in* $\mathcal{A}$ **do**
24   **if** $AR.AS.x > S.xs \wedge AR.AS.x < S.xe$ **then**
25   let $t = \mathsf{clock}()$;
26   let $\mathcal{U} = \mathsf{getHigherPriorityAgents}(AR, t + 2T_M + 2T_D + T_{MAN})$;
27   let $\mathcal{R} = \mathsf{getReachableAgents}(AR, t + 2T_M + 2T_D + T_{MAN})$;
28   **foreach** $turn$ in getPossibleTurns(AR) **do**
29   let $t_{min} = min(\alpha.AS.ta : (\alpha \in \mathcal{A}) \wedge (\alpha.aID \in \mathcal{U}[turn]))$;
30   **if** $\mathcal{U}[turn] \subset \mathcal{R}$ **then** $MR[turn] \leftarrow (t_{min}, \mathsf{True}, \mathcal{U})$ **else** $MR[turn] \leftarrow (t_{min}, \mathsf{False}, \emptyset)$;
31   storeMR($AR.aID, MR$) ;

The Membership Algorithm provides a periodic update of memberships every T_M time units (line 21). The procedure starts by retrieving all agent registries of the vehicles in the selected segment from a storage server (line 22). The algorithm iterates through these agent registers and for each agent p_i checks if p_i is still in the section (line 24). If so, the algorithm calculates which agents $\mathcal{U}$ have higher priority than p_i and may reach the intersection in the time it would take for p_i to reach and perform a manoeuvre through the same intersection (line 26)[vii]. $\mathcal{U}$ will have 3 entries for the 3 possible manoeuvres described above. In addition to $\mathcal{U}$, the set of all vehicles $\mathcal{R}$ that are within communication distance from p_i are determined (line 27). The earliest time stamps t_{min} of the agent states in $\mathcal{U}[turn]$ for each specific turn are also determined (line 29). These time stamps will be used as a measure of how current the memberships are. A t_{min} far back in time indicates that the membership was calculated using old data and that the membership may be too old to be used safely.

The next step is to determine the value of the MOs for the manoeuvre specific memberships. MO is set to True *if and only if* all agents in $\mathcal{U}$ are also in $\mathcal{R}$ (line 30), which implies that all agents that p_i can cause a priority violation for are also reachable by p_i. It's also possible to use the MO-flag to indicate that the provided memberships are invalid due to, for example, limited capacity of the membership service.

The value assigned to $\mathcal{SM}$ depends on the value of MO. $\mathcal{SM}$ is assigned the value of $\mathcal{U}[turn]$ for the specific manoeuvre described by $turn$ if $MO = true$, and is assigned the empty set if $MO = false$ (line 30). The

---

[vii]For more details see Section 3.2.4

Data types:
$MessageType \in \{GET, DENY, GRANT, RELEASE\}$
$StatusType \in \{NORMAL, GET, GRANT, TRYGET, GRANTGET, EXECUTE\}$

Structures:

$AgentState = (ta, x, v, a);$ `// timestamp, position, velocity, acceleration`
$Agent = (\textsf{aID}, AS);$ `// agent ID, AgentState structure`
$AgentTag = (ts, \textsf{aID}, turn);$ `// timestamp, agent ID, intended turn`
$Membership = (tm, MO, \mathcal{SM});$ `// timestamp, MO flag, SM set of agent IDs`
$Message = (senderID, t, type, (data));$ `// agent ID,timestamp,message type,message data`

Constants:
aID: a hard-coded agent identifier

Variables:

$AR \leftarrow \emptyset ;$ `// Agent registry`
$MR \leftarrow \emptyset;$ `// Membership registry, set of Memberships`
$status \leftarrow NORMAL$
$tag \leftarrow \bot;$ `// Unique tag for each manoeuvre`
$grantID \leftarrow \bot;$ `// ID of the agent with an active GRANT`
$\mathcal{M} \leftarrow \emptyset;$ `// Set of received messages`
$\mathcal{D} \leftarrow \emptyset;$ `// Set of destination agent IDs`
$\mathcal{R} \leftarrow \emptyset;$ `// Set of expected response agent IDs`
tRETRY;   `// Timer for keeping track of when to next try starting a new request round`

Interfaces:
clock(): read current time
position(): read current agent position
velocity(): read current agent velocity
acceleration(): read current agent acceleration
startTimer($timer, delay$): start named $timer$ for $delay$ time units
stopTimer($timer$): stop named $timer$
storeAR(AR): write an AR registry to the storage service
getMR(aID): read from the storage service the MR for agent aID
doManoeuvre: execute the manoeuvre by setting the intention to go
noPriorityViolationDynamic($AR, turn$): check if the agent specified in AR have sufficient time to cross the intersection without violating the priority of the ego vehicle, see Section 3.2.5
last(): returns True if $\mathcal{R} \cap \mathcal{D} \cap MR.\mathcal{SM} = \emptyset$, False otherwise
myPriority(m): returns True if $p_{\textsf{aID}}$ has an on going request and that request has the priority over the request of message m, eg having an earlier time stamp, or if equal, lower agent ID.
hasLeftCriticalSection(aID): returns True when the agent p_{aID}'s location is known to be outside the boundaries and heading out from the intersection.
ntfyEstimat(grant, aID/revoke): Notify the risk estimator about priority changes of either a grant of agent aID or the revoke of a grant

Figure 5.1: Structures, variables and interfaces for the manoeuvre negotiation protocol

use of MO thereby provides a way to both reduce the number of sent messages – since vehicles will not send a request if $MO = false$ – and a way to indicate if an empty membership is valid.

Finally, all memberships in the membership set are stored on the storage server accessible by all agents (line 31).

Algorithm 3: The Manoeuvre Negotiation protocol

---

1 **do forever (once in every T_A time units) begin**
2     $AR \leftarrow (\text{aID}, (\text{clock}(), \text{position}(), \text{velocity}(), \text{acceleration}()))$;
3   storeAR(AR);
4     $MR \leftarrow$ getMR(aID) ;
5     $\mathcal{M} \leftarrow \{m \in \mathcal{M} : m.senderID \in (\mathcal{D} \cap MR.SM)\}$;
6   **if** $status = GET \wedge$ last() **then**   `// All expected answers received`
7   stopTimer(tRETRY);   `// No need for a retry`
8   **if** $\nexists m \in \mathcal{M} : m.type = DENY$ **then** $status \leftarrow EXECUTE$; ntfyEstimat(grant) ;
9   **else** $\{status \leftarrow TRYGET;$ **multicast** $\langle \text{aID}, \text{clock}(), RELEASE, (AR, tag) \rangle$ **to** $\mathcal{D}$; startTimer(tRETRY, T_A);$\}$;
10   **if** $status = EXECUTE \wedge$ hasLeftCriticalSection(aID, $MR.\mathcal{SM}$) **then**
11   **multicast** $\langle \text{aID}, \text{clock}(), RELEASE, (AR, tag) \rangle$ **to** $\mathcal{D}$;   `// Explicit release`
12        $status \leftarrow NORMAL$; ntfyEstimat(revoke);
13   **if** $status \in GRANT, GRANTGET \wedge$ hasLeftCriticalSection($grantID, MR.\mathcal{SM}$) **then**
14   ntfyEstimat(revoke); $grantID \leftarrow \bot$;
15   **if** $status = GRANTGET$ **then** $status \leftarrow TRYGET$; tryManoeuvre($tag.turn$);
16   **else if** $status = GRANT$ **then** $status \leftarrow NORMAL$;

17 **procedure tryManoeuvre($turn$) begin**
18   **if** $status \in \{NORMAL, GRANT\}$ **then** $tag \leftarrow (\text{clock}(), \text{aID}, turn)$;
19   **if** $status \in \{NORMAL, TRYGET\}$ **then**
20   **let** $ts = \text{clock}()$;
21        $(status, AR, MR) \leftarrow (TRYGET, (\text{aID}, (ts, \text{position}(), \text{velocity}(), \text{acceleration}())), \text{getMR}(\text{aID})[turn])$;
22   **if** $(ts < MR.tm + 2T_M \wedge MR.MO)$ **then**
23           $(\mathcal{M}, \mathcal{D}, \mathcal{R}, status) \leftarrow (\bot, MR.\mathcal{SM}, MR.\mathcal{SM}, GET)$;
24   **multicast** $\langle \text{aID}, ts, GET, (AR, tag) \rangle$ **to** $\mathcal{D}$;
25   startTimer(tRETRY, $2T_D$);
26   **else** startTimer(tRETRY, T_A);
27   **else if** $status = GRANT$ **then** $status \leftarrow GRANTGET$;

28 **upon timer tRETRY expires begin**
29   **if** $status = GET$ **then**
30        $status \leftarrow TRYGET$; **multicast** $\langle \text{aID}, \text{clock}(), RELEASE, (AR, tag) \rangle$ **to** $\mathcal{D}$; startTimer(tRETRY, $2T_D$);
31   tryManoeuvre($tag.turn$);

32 **upon message m received at time tr begin**
33   **if** $tr - m.t \leq T_D$ **then**   `// Message is timely`
34   **if** $m.type \in \{GRANT, DENY\} \wedge status = GET$ **then** $(\mathcal{M}, \mathcal{R}) \leftarrow (\mathcal{M} \cup \{m\}, \mathcal{R} \setminus \{m.AR.aID\})$;
35   **else if** $m.type = GET$ **then**
36   **if** noPriorityViolationDynamic($requestee, m.AR) \wedge (status \in \{NORMAL, TRYGET\} \vee (status \in \{GRANT, GRANTGET\} \wedge grantID = m.AR.aID) \vee (status = GET \wedge \neg$myPriority($m$)))** then**
37   **if** $status = NORMAL$ **then** $(status, grantID) \leftarrow (GRANT, m.AR.\text{aID})$;
38   **else if** $status \in \{GET, TRYGET\}$ **then**
39   stopTimer(tRETRY);
40   **if** $status = GET$ **then multicast** $\langle \text{aID}, \text{clock}(), RELEASE, (AR, tag) \rangle$ **to** $\mathcal{D}$;
41                 $(status, grantID) \leftarrow (GRANTGET, m.AR.\text{aID})$;
42   ntfyEstimat(grant, $m.AR.\text{aID}$); **send** $\langle \text{aID}, tr, GRANT, \bot \rangle$ **to** $m.AR.\text{aID}$;
43   **else send** $\langle \text{aID}, tr, DENY, \bot \rangle$ **to** $m.AR.\text{aID}$;
44   **else if** $m.type = RELEASE \wedge status \in \{GRANT, GRANTGET\} \wedge grantID = m.AR.\text{aID}$ **then**
45   ntfyEstimat(revoke); $grantID \leftarrow \bot$;
46   **if** $status = GRANT$ **then** $status \leftarrow NORMAL$;
47   **if** $status = GRANTGET$ **then** $status \leftarrow TRYGET$; tryManoeuvre($tag.turn$) ;

5.1.2 Description of the Manoeuvre Negotiation Algorithm

The **Manoeuvre Negotiation Algorithm** (Algorithm 3) consists of four main components: the core state-update procedure that is called periodically (lines 1 to 12), a procedure that is called when an agent seeks permission to perform a manoeuvre (lines 17 to 27), a timer-triggered procedure used to ensure algorithm progression in the event of network failures (lines 28 to 31), and a message processing procedure(lines 32 to

47).

The **state-update procedure** itself has four main parts, the first of which handles communication with the storage server. Every vehicle in our system measures their current position, heading, velocity, and acceleration every T_A time units (line 2) and then stores that information on the storage server (line 3). This vehicle information is used by the membership service to calculate memberships in Algorithm 2. Every vehicle also fetches the latest version of their membership set from the storage server and stores it locally (line 4).

The second part of the procedure determines if a vehicle that is requesting a permission to perform a manoeuvre – a *requester* – has been fully granted. Replies from vehicles that are not a part of the intersection between the membership that was valid when the request was initiated ($\mathcal{D}$) and the current membership ($MR.\mathcal{SM}$) are filtered out from the set of received replies $\mathcal{M}$ (line 5). Vehicles that in $\mathcal{D}$ but not $MR.\mathcal{SM}$ must have already left the intersection, so their replies are considered irrelevant. Vehicles that are in $MR.\mathcal{SM}$ but not $\mathcal{D}$ are considered too far away to have any impact on the requested manoeuvre. Vehicles in neither set are also either too far away, or have at most equal priority to the requester and therefore should not be allowed to affect the decision process.

Next, the requesters – having the status GET – confirm that all expected replies to the sent request have arrived by calling $last()$ (line 6). If all expected replies have arrived then the tRETRY is stopped to mark the end of the current request round (line 7). In order to assess if the requester is fully granted, $\mathcal{M}$ is searched for any $DENY$ replies and, if none are found, the requester changes status to $EXECUTE$ and is thereby free to enter the intersection (line 8). Otherwise the requester will send a $RELEASE$ message to all vehicles in $\mathcal{D}$ telling them to stop considering the requester granted and the tRETRY timer will be started again to continue the request into a new round (line 9).

The third part of the procedure determines when a fully granted vehicle should release its grant. A fully granted agent p_i – having status $EXECUTE$ – will use their measured position and knowledge regarding the intersection's geometry to determine if it has left the intersection (line 10) and should therefore send a $\langle RELEASE \rangle$ of the grant to every vehicle in $\mathcal{D}$ (line 10). The agent p_i will then return to status $NORMAL$ and call ntfyEstimat(revoke) to also inform the RE about the revoked grant (line 12).

The last part of the state-update procedure handles implicit release of a grant which enables a grant to be released even when the explicit release message is lost. An agent that is currently granting another agent, having status $GRANT$ or $GRANTGET$, checks if the latest measured position of the granted agent lies outside and is heading away from the intersection (line 13). If the condition is True then the agent notifies the RE about the priority change by calling ntfyEstimat(revoke) and clears $grantID$ (14). An agent previously prevented from starting a request round, denoted by having status $GRANTGET$, takes on status $TRYGET$ and calls tryManoeuvre to try to start a new request round (line 15). An agent with status $GRANT$ instead goes back to status $NORMAL$ (line 16).

The request initiating procedure tryManoeuvre is invoked when starting a new round of a requests. During the first round of a request, where the vehicle has status $NORMAL$ or $GRANT$, the procedure generates an identification tag for the request, then stores: the time stamp of when the first request round started, the ID of the requester, and the manoeuvre the requester wants to perform (line 18). Requesters that are not currently granting another agent, having status $NORMAL$ or $TRYGET$, proceed to measure their current agent state and fetch the latest membership from the storage server (line 21).

Starting a request round further requires a valid membership, implying that the membership is based on data no older than $2T_M$ time units and with a $MO = true$ (line 22). If the membership is valid, then the state variables $\mathcal{M}$, $\mathcal{D}$, and $\mathcal{R}$ will be assigned the values $\perp$, $MR.\mathcal{SM}$, and $MR.\mathcal{SM}$ respectively, where $\mathcal{R}$ is the set of agents that are expected to reply and the status is changed to GET (line 23). A $\langle GET \rangle$ request is then sent to all agents in $\mathcal{D}$ along with the request tag. The tRETRY timer is set to $2T_D$ to limit the duration of the request round (line 25). An agent without a valid membership will wait for a new membership from the membership server by starting the tRETRY timer and try starting a request round at a later point in time 26. A potential requester that is currently granting another agent, having status $GRANT$, is not allowed to start a request round until the grant is released, but a status change to $GRANTGET$ is made to signify that the agent wants to send out a request later when it is allowed (line 27).

Upon expiry of the tRETRY timer, an agent currently in a request round, having status *GET*, will end it by sending a *RELEASE* to all agents in $\mathcal{D}$, changing its status to *TRYGET* and starting the tRETRY timer again (line 30 and 30). The agent did not get all expected replies in the set time frame and must therefore stop the current round to avoid deadlocks. If the timer expires for an agent without an ongoing request round, having status *TRYGET*, then the agent will try to start a new request round by calling the tryManoeuvre procedure (line 31).

The final part of the Manoeuvre Negotiating algorithm determines how incoming messages are processed depending on the agent's current status and the agent state of the sender. Messages sent more than T_D time units ago are discarded in order to avoid messages from previous request rounds interfering with the current round (line 33).

The first type of messages that are processed are request replies on the form $\langle GRANT \rangle$ and $\langle DENY \rangle$. Any requester will store these messages in the set $\mathcal{M}$ and remove the sender ID from the set of agents whose expected responses have not yet arrived, $\mathcal{R}$, (line 34).

The second message type that is processed is $\langle GET \rangle$ which symbolises a permission request (line 35). The receivers have to decide if the request should be granted or denied. Firstly, a call to noPriorityViolation determines if the sender can, while following normal driving patterns, cross the intersection without blocking the receiver (line 36). In addition to the non-blocking criteria, the receiver has to check its own status to see if it allows the agent to grant the sender.

Status *NORMAL* and *TRYGET* imply that the agent does not have an active request round and is therefore allowed to grant another agent (line 36).

Agents with status *GRANT* or *GRANTGET* are currently holding a grant for another agent. If this is the same agent as the sender then the receiver is allowed to reply with a $\langle GRANT \rangle$ (line 36).

The last status that allows the receiver to reply with a grant occurs where the receiver both has status *GET* – meaning that it has an active request round – and the sender has higher priority than the receiver – determined by a call to the interface myPriority (line 36). The interface myPriority will return False if either the sender's request was started before the receiver's request or if they were started concurrently and the sender's ID is lower than the receiver's.

A receiver fulfilling both the non-blocking and status criteria will store the ID of the granted agent if it has not yet done so, and change its status to either *GRANT* – if it has status *NORMAL* – or to *GRANTGET* – if it has status *GET* or *TRYGET* (lines 37, 38, and 41). Changing from status *GET* or *TRYGET* to *GRANTGET* also implies both that the tRETRY timer has to be stopped and that any ongoing round must be stopped by sending a $\langle RELEASE \rangle$ to all agents in $\mathcal{D}$, since an agent in *GRANTGET* is not allowed to either start or have an ongoing request round (lines 39 and 40). The risk estimator is then notified with ntfyEstimat(grant) that the priorities have changed and a $\langle GRANT \rangle$ reply is sent to the request sender (line 42). If these criteria are not fulfilled, a $\langle DENY \rangle$ is sent in reply (line 43).

The last type of messages to be processed are $\langle RELEASE \rangle$ messages. An agent with status *GRANT* or *GRANTGET* that receives a release message from the holder of the grant will call ntfyEstimat(revoke) to inform the risk estimator of the expiry of the granted vehicle's higher priority (45). A *GRANT* agent will return to status *NORMAL* while a *GRANTGET* agent switches to status *TRYGET* and calls tryManoeuvre in order to try to start a request round (line 46 to 47).

5.2 Correctness Proof

By the stature of Algorithm 3, we can observe the correctness of Corollary 5.1.

Corollary 5.1 *Let E be an execution of the proposed solution. At any system state, it holds that ($status \in \{TRYGET\} \implies active(\text{tRETRY})$), where $active(t)$ is a predicate that represents the fact that timer t has been started (by the function startTimer(t)), and has not stopped (by the function stopTimer(t)) or expired without triggering the event bounded to timer t. Furthermore, an active tRETRY timer expires within $B\_RETRY = \max\{2T_D, T_A\}$ time units.*

Lemma 5.2 (Membership) *Let E be a synchronous execution of algorithms 2 and 3. Requirement 4.1 holds throughout E.*

Proof. To prove that Requirement 4.1 holds throughout E it needs to be shown that a fresh membership M of an agent p_i with $MO = \mathsf{True}$ contains *all* agents that could potentially cause a priority violation with p_i from the current time T until $T + 2T_D + T_{MAN}$. We say that a membership is *valid* if this property holds for the membership. We refer to all other memberships as non-valid. It needs to be proven that, when t is the timestamp bound to M (line 29), M can only be non-valid if $MO = \mathsf{False}$ or M is no longer fresh, i.e., $t \geq T - 2T_M$.

Algorithm 2 (lines 21 to 31) defines the procedure in which memberships are created and stored. We review this procedure and show its key properties.

Algorithm 2 considers all potential priority violations. The procedure calls $\mathsf{noPriorityViolationStrict}(t + 2T_M + 2T_D + T_{MAN})$ to determine all agents $\mathcal{U}$ that have higher priority than p_i and that could potentially reach the intersection at some point from t to $t + 2T_M + 2T_D + T_{Man}$ (line 26). These agents $\mathcal{U}$ are the ones that could potentially lead to a priority violation with p_i according to our definition in Section 3.2.3.

Algorithm 2 assigns correct MO values. For a given turn, the membership M is assigned the set $\mathcal{U}[turn]$ if, and only if, MO is assigned the value True, otherwise M is assigned the empty set (line 30). The variable MO thereby distinguishes a valid empty membership set from a non-valid empty membership set.

Algorithm 2 assigns correct time stamps. The time stamp associated with a membership is assigned the value of the earliest time stamp t_{min} of the agents in $\mathcal{U}[turn]$ (line 29). This implies that a fresh membership is valid from T to $T + 2T_D + 2T_M + T_{MAN} - (T - t_{min})$ where $T - t_{min} \leq 2T_M$ due to the freshness of M. This proves Requirement 4.1 since we assume that vehicles do not drive backwards on highways. ∎

Theorem 5.3 (Safety) *Let E be an execution of algorithms 2 and 3 that is not necessarily synchronous in which Requirement 4.3 holds throughout. Requirements 4.2, 4.3, and 4.5 hold throughout E.*

Proof.

1. **Requirement 4.2:** By the structure of Algorithm 3, we can observe that an agent p_i that aims to perform a manoeuvre, has to call $\mathsf{tryManoeuvre}()$ (line 17) and be granted a permission before reaching the status $EXECUTE$. In detail, $\mathsf{tryManoeuvre}()$ is the only interface procedure for agent p_i, which includes lines 23 to 25 in which requests are sent. By line 22, we see that p_i sends such a request only when (i) its membership is fresh and (ii) there is an positive indication for a manoeuvre opportunity. Otherwise, p_i checks conditions (i) to (iii) after T_A time units (line 26). Thus, Requirement 4.2 is fulfilled.

2. **Requirement 4.3:**

   Algorithm 3 defines how a $\langle GET \rangle$ message is sent (lines 22 to 25) and received (line 35 to 43). We note that p_i sends a $\langle GET \rangle$ message only if the if-statement condition at line 36, i.e., ($t_i < M.tm + 2T_M \wedge MR.MO$) which implies that the first assumption of Requirement 4.3 holds. To show that the second assumption holds, we note that p_j processes a request message $\langle GET \rangle$ from p_i at a time $t_j \in [t_i, t_i + T_D]$ only when its delivery is timely (line 33). Thus, we only need to prove that a request sent by a requester p_i at a time t_i to a requestee p_j in p_i's membership M is replied to with a $\langle GRANT \rangle$ if, and only if, at a time $t_j \in [t_i, t_i + T_D]$ when the request is delivered to p_j, $\mathsf{noPriorityViolationDynamic}(AR_j) = \mathsf{True}$ and p_j is in one of the states presented in the cases (i) to (iii) described in Requirement 4.3.

   From line 36 we observe that $\mathsf{noPriorityViolationDynamic}(AR_j) = \mathsf{True}$ is a mandatory condition for a request to be granted. From line 36, we can also observe that if (a) p_j has status $NORMAL$ or $TRYGET$, (b) p_j's status is $GRANT$ and the granted agent is p_i, or (c) p_j's status is GET and p_i has higher priority according to $\mathsf{myPriority}$, then p_j will send a $\langle GRANT \rangle$ and call $\mathsf{ntfyEstimat}(grant)$ (line 42). Moreover, p_j reply with a $\langle DENY \rangle$ if p_j is in a state different from cases (a) to (c) or if $\mathsf{noPriorityViolationDynamic}() = False$ (line 43). The cases (a) to (c) indeed correspond to (i) to (iii), which Requirement 4.3 defines. Since the lines 42 and 43 are the only locations in the algorithm where a $\langle GRANT \rangle$ or $\langle DENY \rangle$ is sent, Requirement 4.3 is satisfied.

22

3. **Requirement 4.4:**

   From Algorithm 3 we see that a $\langle RELEASE \rangle$ message is sent to all agents in $\mathcal{D}$ only on the lines 9, 30, 12, and 45. We need to show that these lines are reached if, and only if, any of the cases (i) to (iv) presented in Requirement 4.4 hold. We further need to show that ntfyEstimat(revoke) is only called when an agent p_j receives a $\langle RELEASE \rangle$ from another agent p_i and p_j is currently holding a grant for p_i or when p_j is in state $EXECUTE$ and has left the intersection.

   To reach and execute line 9, the requesting agent p_i (with status GET) needs to receive replies before the time $T = t_i + 2T_D - T_A$, where t_i is the time when p_i's request was sent, otherwise the tRETRY timer will expire before the if-state condition on line 6 evaluates to True. The agent p_i's tRETRY timer will be stopped immediately after p_i discover that all needed replies have arrived to stop the timer from interfering (line 7). Furthermore, any received $\langle DENY \rangle$ message before time T from another agent p_j will be stored by p_i (line 34). However, p_j's $\langle DENY \rangle$ will only be kept if $p_j \in (\mathcal{D} \cap M)$, where M is p_i's current membership (line 5). The agent p_i will then execute line 9 since p_j's $\langle DENY \rangle \in \mathcal{M}$ (line 8). This case correspond exactly to case (i).

   If not all messages arrive before time T, then the tRETRY timer set by p_i when sending the request (line 25) expires and p_i will execute line 28. The agent p_i will then reach and execute line 30 since p_i has status GET. This corresponds to case (ii). An agent cannot reach line 30 in any other way since all other cases in which p_i starts the tRETRY timer (lines 9, 26, and 30) the agent will have status $TRYGET$.

   To execute line 12, agent p_i's status has to be $EXECUTE$ and p_i has to have left the intersection so that $hasLeftCriticalSection(p_i, M)$ evaluates to True (line 10). This correspond to case (iii).

   The last line in which a $\langle RELEASE \rangle$ can be sent, line 45, can be reached only if an agent p_i has status GET when receiving a $\langle GET \rangle$ message m from another agent p_j and if $\neg$myPriority$(m) = $ True, and noPriorityViolationDynamic$(AR_j) = True$ (lines 35, 36, 38, and 40). This corresponds to case (iv).

   We consider the case in which agent p_j has the status $GRANT$ and holding a grant for p_i while receiving a $\langle RELEASE \rangle$ message from p_i. In this case, p_j calls ntfyEstimat(*revoke*) (line 12). The only other line where
   ntfyEstimat(*revoke*) is called is line 14. This line can only be reached if an agent has status $EXECUTE$ and if hasLeftCriticalSection(aID) evaluates to True (line 10). Requirement 4.4 is thus satisfied.

4. **Requirement 4.5:** Note that p_i does not remove messages that are needed (such as in line 5). By line 6 to 9 and the definition of the function last() in Figure 5.1, agent p_i does not change its status to $EXECUTE$ before it receives all the needed responses. This change to $EXECUTE$ depends on the absence of a response that includes a $\langle DENY \rangle$, i.e., the case of being fully-granted. Otherwise, p_i changes *status* to $TRYGET$, sends a release message to all agents in $\mathcal{D}$ and restarts the request process after T_A time units.

   ∎

Theorem 5.4 (Progression) *Let E be a synchronous execution of algorithms 2 and 3 in which admissible crossing opportunities occur infinitely often (for every direction of the intersection). Requirement 4.6 holds throughout E.*

 Proof. Let p_i be an agent that is the closest to the intersection from its direction and that aims at crossing the intersection, i.e., it calls tryManoeuvre() (line 17). By lines 21 and 27, immediately after the execution of tryManoeuvre(), the state of p_i cannot be in $\{NORMAL, GRANT\}$. This proof shows that the states GET, $TRYGET$, and $GRANTGET$ must lead to $EXECUTE$ before returning to the state $NORMAL$ by demonstrating that p_i indeed reaches the status $EXECUTE$ and crosses the intersection.
Without loss of generality, suppose that E is an execution in which p_i's status is not in $\{NORMAL, GRANT\}$ during the first system state of E. Moreover, suppose that immediately after E's starting system state, we have a period of $T_{admissible} > B\_RETRY + 2T_A$ time units in which p_i has an admissible crossing

opportunity, where $B\_RETRY = \max\{2T_D, T_A\}$. Generality is not lost due to the statement in the beginning of this proof and because Requirement 4.6's assumptions, which means that an admissible crossing opportunity for p_i always occurs eventually (within a finite period).

The rest of the proof considers p_i's status in E's starting system state, which can be one of GET, $TRYGET$, and $GRANTGET$, and shows that it indeed reaches status $EXECUTE$.

- **Suppose that status$_i$ = GET.** Note that the do forever loop (line 1) of Algorithm 3 starts a new iteration once in every T_A time units. Therefore, within T_A time units, agent p_i executes line 6. Once that occurs, the if-statement condition in line 6 holds, due to the assumption of this case that the status of p_i is $= GET$, the definition of the function last() (Figure 5.1) and the assumption that during the first $T_{admissible}$ time units of E it holds that p_i has an admissible crossing opportunity, i.e., $MR_i.\mathcal{SM} = \emptyset$ and $MR_i.\mathcal{SM}$ is fresh.

  Note that the removal of stale information in line 5 implies that $\nexists m \in \mathcal{M} : m.type = DENY$ holds. This means that the if-statement condition in line 8 holds and p_i gets status $EXECUTE$. Thus, the proof is done for this case, because $T_{admissible} > 2T_A$.

- **Suppose that status$_i$ = TRYGET.** By Corollary 5.1, it holds that $status = TRYGET$ implies that the timer tRETRY is active. Thus, within $B\_RETRY = \max\{2T_D, T_A\}$ time units agent p_i executes the procedure associated with this timer expiration (line 28), which calls the procedure tryManoeuvre($tag.turn$) (line 31). During the execution of the procedure (line 17), it holds that the if-statement condition in line 19 holds (due to this case's assumption that $status = TRYGET$). By the assumption that during the first $T_{admissible}$ time units there is an admissible crossing opportunity in E with respect to agent p_i, it holds that the if-statement condition in line 22 evaluates to True. Once that happens, line 23 assigns GET to $status_i$ before setting the timer tRETRY to a value larger than $2T_A$ (line 25). Thus, p_i start and completes the execution of the do forever loop (line 1 to 12) with $status_i = GET$ before the expiration of the timer tRETRY (line 28). The proof is thereby done for this case by arguments similar to the case of $status_i = GET$, since $T_{admissible} > B\_RETRY + 2T_A$.

- **Suppose that status$_i$ = GRANTGET.** By the definition of an admissible crossing opportunity, p_i's membership will be empty for at least $T_{admissible} > B\_RETRY + 2T_A$ time units. An empty membership implies that no other agents can cause a priority violation with p_i, which in turn implies that the agent p_j that p_i is granting must have left the intersection. Within T_A time units, p_i evaluates the if-statement in line 13 to True, revokes the held grant, changes status to $TRYGET$, and calls tryManoeuvre (line 14 to 15). The rest of the proof is identical to the latter part of the $TRYGET$ case above.

∎

6 Implementation

6.1 The Risk Estimator

The **Risk Estimator** (RE) used in the simulations is based on [11], and was modified in order to utilise *speed models* – as opposed to *constant speed* – when making vehicle motion predictions.

The RE represents every vehicle in the traffic scene with a *particle filter*. Each **particle filter** has weighted particles, where each particle has a position, heading, speed, manoeuvre intention, and expectation and intention to go or stop, representing the state of the vehicle. These particles are projected one time step forward in time based on their current *state vector* using simple motion formulas (position, heading, speed) and a *Hidden Markov Model* (HMM) (Expectation and Intentions). New weights are then assigned to the particles based on their similarity to the last received AS from the respective vehicle. A multivariate normal distribution is used as a likelihood function to measure similarity and the measured entities are assumed to be independent.

After the batch of particles for every vehicle have been projected and re-weighted, the risk estimator then estimates the new expectations. Expectation is based on the traffic priority rules and time gaps of when the vehicles are estimated to enter the intersection, we refer to this time as **Time To Entry** (TTE). Speed models for every manoeuvre and intention to go or stop are used to estimate TTEs. For non-priority vehicles, the difference between the TTE for each vehicle is fed into a gap model, which determines if the gap is large enough for the vehicle to be expected to go, or whether it will be required to stop and wait for higher priority vehicles. The cumulative distribution function for the gap model is presented in Equation (6.1),

$$Y = \frac{1}{1 + e^{b(a-x)}}. \tag{6.1}$$

where the constants a and b can be adjusted to change the *threshold position* and *steepness* of which time gaps are considered big enough with a high likelihood. Priority vehicles will have the expectation to go.
As mentioned previously, the *manoeuvre intention* and *intention to go or stop* is determined by a HMM, see Figure 6.1. The HMM is designed let a vehicle change the manoeuvre intention with a probability of 10 % in every time step, representing the fact that most drivers keep to their manoeuvre plan. Drivers are also modelled as usually adhering to the expectations, with a chance of 90 % to comply if the current expectation matches the previous intention and a chance of 50 % if they do not match.
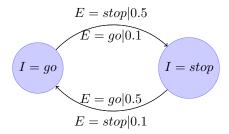

Figure 6.1: Visual representation of the HMM for the intention to go or stop. The transitions, labelled $a|b$, represent the likelihood b of changing to the other value if the expectation is calculated to be a.

In the beginning of a new round of estimations, re-sampling will occur if there are too few particles with a high enough weight. Particles with a weight under a certain threshold will then be discarded and more likely particles will be duplicated. This process is called *importance sampling* and is used to reduce the risk of *sample depletion* when almost all particles have negligible weights, while still minimising the amount of lost information [8]. If all weights equal 0 after an estimation round, the particle filter is restarted with new particles spread around the latest received agent state.
Finally, the risk is computed for a vehicle by computing the sum of the normalised weights of the particles representing it where the intention is go but the expectation is to stop. A sum higher than the set risk threshold triggers an **emergency break** (EB) in which the vehicle sets its intention to stop.
Having an intention to stop but expectation to go may also be considered risky when multiple vehicles are driving in a line but we decided to not include it in our calculations at this stage since we are studying collisions caused by vehicles coming from different directions.

6.2   Internal hierarchy for deciding vehicle intention

Both the Manoeuvre Negotiation protocol and the Risk Estimator influence a vehicle's intention to go or stop. According to the Manoeuvre Negotiation Protocol, all vehicles should have the intention to stop until they are fully granted. The Risk Estimator, on the other hand, sets a vehicle's intention based on a gap model to calculate the expectation and a HMM to, based on the previous intention and the current expectation, calculate the intention. The Risk Estimator also calculates a risk of a vehicle's intention not matching the expectation and triggers an emergency break if this risk passes a set threshold. Emergency break in this case

is the same as setting the intention to stop. In addition to our safety system, traffic rules can also be used to set a vehicles intention. There are therefore 4 possible sources of information a vehicle can use to set its intention and setting up an internal hierarchy between them is necessary to avoid a arbitrary behaviour. The hierarchical order we decided upon with the most influential indicator first is

1. Emergency break

2. Priorities set by the Manoeuvre Negotiation Protocol

3. Risk Estimator's HMM

4. Priorities determined by traffic rules

An emergency break will set the intention of a vehicle to stop no matter what the other system's say. If no emergency break is triggered then the priorities set by the Manoeuvre Negotiation Protocol decide if a vehicle should go or stop. Lastly, if no emergency break is triggered and the manoeuvre negotiation protocol is not in use, the output from the HMM decides the intention of a vehicle.

If none of our safety systems are active then the traffic rules themselves can be used to set the intention. The traffic rules are placed last on the list but they influence the decisions made by both the Risk Estimator and the Manoeuvre Negotiation Protocol.

6.3 Architecture

The architecture setup can be found in Figure 6.2. An intersection traffic scenario, a membership service and two cars – each following the manoeuvre negotiation protocol and having their own risk estimator – were simulated using ROS. A storage server for vehicle state data and memberships was implemented as an Apache Zookeeper server.

6.3.1 Storage server

The manoeuvre negotiation protocol relies on a storage server where agents regularly upload their current state and read their memberships computed by a membership service, here referred to as *the cloud*. In our implementation, this storage server was chosen to be an Apache Zookeeper server because it provides a framework for building a highly reliable and distributed server. Communication to the server was done via TCP.

The Zookeeper server has 2 main directories (called znodes): *segment* and *mr*. Segment stores the *agent states* (time stamp, position, heading, speed) that all vehicles regularly update and that the cloud uses to compute memberships. The memberships come in sets, with one membership per possible manoeuvre through the intersection. Each membership is stored in *mr* under the ID for every vehicle and has the form (Turn, Manoeuvre opportunity, Safety membership)[viii].

The vehicles periodically write their current state to the storage server and read their own membership set, whilst the cloud in turn reads the agents' agent states and then writes their computed memberships to the server (see Figure 6.2).

6.3.2 Simulation environment

Every vehicle is represented by a ROS node in the simulation environment and the vehicles publish their Agent State (AS) every T_A time units to their own ROS topic (a publish-subscribe communication channel in ROS). The vehicles subscribe to other vehicle's topics and use the received states as inputs to their risk estimator. Every vehicle further sends their AS and retrieves their membership set from the storage server over TCP. The messages sent in the manoeuvre negotiation protocol uses UDP since the protocol itself is designed to handle packet loss and a much delayed (possibly retransmitted) packets are discarded in the protocol anyway.

[viii]See Section 5.1.1 for a more detailed explanation of the parts of a membership
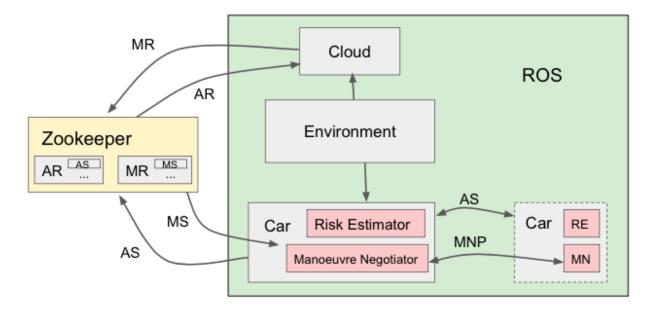
Figure 6.2: A high-level map of the simulation environment. One or two-headed arrows designate communication channels that are mono-respective bi-directional. Both the vehicles and the cloud were set up in ROS while a Zookeeper server was used as an external storage server. The environment block represents information regarding the geometry of the intersection.

The membership service is also implemented on a ROS node. It reads all the agent states (AR) from the *segment* znode on the storage server and then writes all the computed memberships (MR) to another znode, *mr*, on the same storage server. The memberships are computed using the ASs and intersection information, which includes the priorities and geometry set up at when the simulation was initiated.

6.3.3 The Risk Estimator

Every simulated vehicle was equipped with a **Risk Estimator**. The risk estimator chosen was based on one developed by by [11] but which some modifications[ix]. The risk estimator was run as a background process throughout the simulation, as explained in further detail in Section 3.4.3. Each particle filter used by the RE was set to have 625 weighted particles representing the state of a vehicle.

6.3.4 The Manoeuvre Negotiation Protocol

Every vehicle node in ROS was equipped with an object to handle the Manoeuvre Negotiation Protocol in Algorithm 3. Another node, the *Cloud*, was dedicated to running the Membership Algorithm in Algorithm 2 in which memberships for every vehicle in the traffic scene are computed.
More details on the Manoeuvre Negotiation Protocol are provided as line by line descriptions in Section 5.1; a correctness proof is given in Section 5.2.

7   Evaluation

This chapter describes the different test scenarios, evaluation criteria and how tests were compared to validate our implementation of the risk estimator and the manoeuvre negotiation algorithm. Tests were also

---

[ix]see Section 6.1

conducted to show liveliness of the updated protocol.

7.1 Simulation environment

A 4-way intersection was set up in ROS with predefined paths for every incoming directions' manoeuvres, corresponding to turning left or right or going straight. The north-south lanes were given higher priority in respect to the crossing east-west lanes. The simulated vehicles were implemented on ROS nodes. A speed model decided the vehicles' speed depending on their distance from the intersection, see Figure 3.3. A *PID-controller* adjusted the steering angle to keep the vehicles on the paths. The vehicles exchanged state information with added Gaussian noise every 0.5 s using ROS-topics. The manoeuvre negotiation was conducted using UDP while the communication with the storage server used TCP. Every vehicle had their own instance of a risk estimator and manoeuvre negotiator.

7.1.1 Simulation scenario

A great percentage of all traffic accidents with serious or fatal outcomes occur in intersections. Of all the possible turns through intersections, left-turn-across-path is considered one of the most collision-prone which is why we have decided to focus on this manoeuvre in our tests. The simulations involve two vehicles approaching an unsignaled intersection from opposite directions, where one vehicle **intends to do a left turn**, V_L, while the other **intends to go straight**, V_H, see Figure 7.1. Both vehicles are on the priority lane but they do not have equal priority in the intersection since intending to doing a left turn initially decreases a vehicle's priority. The test cases will therefore involve V_L requesting permission from V_H to change the priorities.
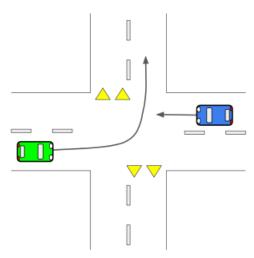

Figure 7.1: The scenario of a vehicle, V_L, turning left across the priority lane with an opposing priority vehicle, V_H, approaching, used for testing our system.

7.1.2 System comparisons

To evaluate our contribution to the Manoeuvre Negotiation protocol we want to compare its performance to tests conducted with the risk estimator alone and with the two systems combined. These tests are needed to check if the combined system performs better than any of the systems by itself. Values that will be compared are explained further in Section 7.2. Reference values without emergency break and manoeuvre negotiation were calculated from the starting position in every case instead of simulating their runs since these vehicles

deterministically following the speed model without stopping. The following setups were tested for every test case

- **RE**: Running the risk estimator alone with the emergency break feature on

- **MN**: Using the Manoeuvre Negotiation protocol alone, no emergency break

- **RE+MN**: Running the risk estimator with emergency break and the Manoeuvre Negotiator

Each of these setups were run with the same list of seeds for the random number generator to give the runs the same initial conditions. The seed affects the risk estimator – which uses random numbers in the particle filter – and the manoeuvre negotiation, which depends on a random function when a received request is processed. Every test case was run 10 times with 2 different seeds each time (one for each of the two libraries of random functions used). The quantitative evaluation criteria were computed by taking a mean of the 10 values with the highest and lowest value removed. The qualitative criteria were calculated as a mean over all 10 values, which results in a number between 0 to 1 indicating how likely an event is when certain start values are deployed.

7.2 Evaluation criteria

The tests were evaluated based on criteria that measure the:

- Number of collisions and dangerous situations. A **collision** is here defined as a case where the vehicles both occupy the space shared by both their paths and a **situation** is defined as *dangerous* if the distance separating the front of each vehicle is less than $4\,m$. Saved state data from the simulations was used to calculate these values.

- **Time Lost due to Priority Violations** (TLPV). TLPV is only computed for the initially higher priority vehicle V_H and will be positive that vehicle has to slow down due to the actions of the other (lower priority) vehicle V_L. This would occur when, for example, either the emergency break is triggered or if the other vehicle is both granted and then releases the grant later than expected. TLPV is a qualitative measure of the implementation of noPriorityViolation. Time to enter the intersection for each of the test cases were compared with a computed optimal value from a scenario where the priority vehicle does not have to interact with any other vehicle.

- Number of emergency breaks triggered by the risk estimator. The outcomes from the RE test case are used here to determine whether or not adding the Manoeuvre Negotiation Protocol will cause the risk estimator to give false alarms.

- **Time To get fully Granted** (TTG) after the initial request was sent. This measure was recorded to demonstrate liveliness of the Manoeuvre Negotiation Protocol. It also provides a measure of through-put.

Emergency breaks and TLPV can both be signs of priority violations for the higher priority vehicle, caused by the risk estimator and Manoeuvre Negotiation Protocol separately. An emergency break signal will occur if the risk estimator computes that the lower priority vehicle is about to drive through the intersection before the higher priority vehicle when the time gap is considered too small. TLPV instead measures how much the priority vehicle is slowed down by having to break in order to let the granted lower priority vehicle finish its manoeuvre. It is therefore a measure of how well the manoeuvre negotiation protocol approximated the time of the manoeuvre execution for the lower priority vehicle.

To measure the performance of our implementation of the risk estimator we further conducted some tests measuring the:

- **Time To Collision Point** (TTCP) when the risk estimator first triggered an emergency break. This provides a measure of how early the risk estimator was able to detect a possible collision. The TTCP

was computed using the vehicle's position in combination with the speed model shown in Figure 3.3, which produced an estimate as to when the vehicle would first reach the *collision zone* (at the collision point). The collision zone is the part of the two vehicles' trajectories that overlap when also considering the size of the vehicles.

- Precision, which determines if the risk estimator is prone to producing a large number of false alarms. **Precision** is here defined as the number of test cases where an actual dangerous situation was detected, divided by the total number of the cases where the emergency break was triggered.

- Recall, which measures how well the risk estimation is able to detect potential collisions. **Recall** is here defined as the number of test cases where an emergency break was triggered in a dangerous situation divided by the total number of test cases where a dangerous situation occurred.

Observe that TTCP is defined to be a non-negative measure, because any instance where an EB event is triggered and the vehicle is either in the collision zone of has left the collision zone are assigned a TTCP value of 0, since at that point the vehicle has either reached or passed the collision point. It follows that any EB events after this point cannot help in preventing a collision.

7.3   Test Cases

The distance parameters used during the simulation are all measured from the centre of the intersection, see Figure 7.2. The start distance for V_L, d_0, and the request initiation distance, d_{Init}, were kept fixed to $65\,m$ and $30\,m$ respectively. The start distance for V_H, d_1, was varied [x].
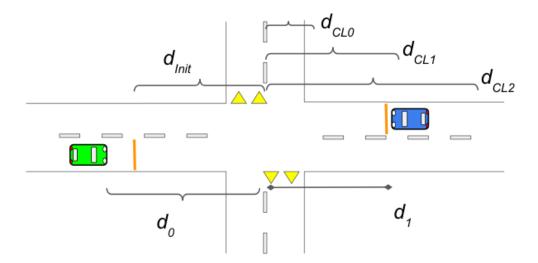

Figure 7.2: Distance parameters used during the simulations. All distances are measured from the middle of the intersection and all except d_1 are fixed. The values for d_{CLX} can be found in Table 7.1, $d_{Init} = 30\,m$, $d_0 = 65\,m$ and the values for d_1 are described in 7.3.

Each test case focused on varying one parameter while keeping the others fixed or, as in the cases with added noise or communication loss, limited to a few values in order to try to determine how varying the parameter affects the results. Here follows a presentation on the different test cases and a motivation as to why they where chosen to evaluate our implementation. The letter combination used to refer to each test case is set between parenthesis in the test case's title.

---

[x]see Section 7.3.1 for more details

7.3.1 Test Case: Normal (Normal)

Variations in starting distance result in different time gaps of when the two vehicles are estimated to enter the intersection and should therefore affect everything that is related to our risk estimation. The start distance d_1 for V_H was therefore varied in steps of $4\,\text{m}$ between 125 to $13\,\text{m}$, measured from the centre of the intersection. The other vehicle's start distance d_0 was kept constant.

7.3.2 Test Case: Noisy Measurements (Noise)

Noise in various levels is always present in real-life measurements. It is therefore of great importance to test how our implementation performs under variously noisy conditions.
In this test case, extra noise was added to the measured agent state values including speed, position and angle before using the values in the system. Since the noise added in the Normal case in Section 7.3.1 was Gaussian with a true mean and standard deviation of σ, we decided to add two test cases here, one with a standard deviation for the noise of 1.5σ and the other of 2σ. For each of these two settings, the different start position runs as described in Section 7.3.1 were performed.

7.3.3 Test Case: Lost communication (ComLoss)

Message loss is also a common fault that occurs in real systems. To simulate message loss in our simulation we added a total lack of communication for V_L at 3 different distances from the intersection d_{CL0}, d_{CL1}, and d_{CL2}, for 3 different time durations t_{CL0}, t_{CL1}, and t_{CL2}. Their values can be found in Table 7.1 and the d_{CLi}-distances are depicted in Figure 7.2. Notice that the three starting values correspond to long before the request line, just before the request line, and just before entering the intersection. For every combination of $d_{comstop}$ and $t_{comstop}$, varied start position for V_H was applied as in Section 7.3.1.

d_{CL} (m)	11	31	51
$t_{comstop}$ (s)	1	2	3

Table 7.1: Values for the communication loss parameters $d_{comstop}$ – determining how far from the intersection communication was cut for V_L– and $t_{comstop}$ – how long communication was cut. These values were combined to form 9 different test cases.

This test should give an indication on how well the systems are able to handle a lack of new information for a limited time period at different distances from the intersection.

7.3.4 Test Case: Traffic Offenders (Offender)

Not all vehicles follow traffic rules and conventions, so a robust system has to prevent traffic accidents even if traffic offenders are present. A traffic offender was simulated by letting V_L ignore priorities while still sending positional messages and other messages required by the protocol. The offender has its intention set to *go* throughout the simulation, which implies that the vehicle drives without following the expectation set by the risk estimator or priorities set by the protocol, and does not respond to triggered emergency breaks. The test runs had the same starting conditions as in Section 7.3.1 with varying start distance d_1. The purpose of the test is to evaluate the risk estimator in an environment where not all vehicles are adhering to their safety system, unlike all previous test cases presented here.

7.4 Evaluation terminology

We have developed the following terminology for ease of reading:

- A particular **Setup** will be denoted $\mathcal{S}_{Type}$. E.G. The *RE* setup will be denoted $\mathcal{S}_{RE}$.

- A particular **Test Case** will be denoted $\mathcal{TC}_{Type}$. E.G. The *Normal* test case will be denoted $\mathcal{TC}_{Normal}$.

- An **Evaluation Scenario** will refer to a combination of one test case $\mathcal{TC}$ and one setup $\mathcal{S}$, and will be denoted $\mathcal{ES}^{Setup}_{TestCase}$. E.G. $\mathcal{TC}_{Normal}$ paired with $\mathcal{S}_{RE+MN}$ will be denoted $\mathcal{ES}^{RE+MN}_{Normal}$.

8   Results

This section presents the results obtained from running the test cases described in Section 7.3 and measuring the evaluation criteria defined in Section 7.2.

Our initial hypothesis was that $\mathcal{S}_{RE+MN}$ would prove to be a more safe system than $\mathcal{S}_{RE}$, which was confirmed by the results: there were fewer collisions and dangerous situations recorded when using $\mathcal{S}_{RE+MN}$. This system also reduced the amount of triggered emergency breaks compared to $\mathcal{S}_{RE}$, while still keeping TLPV low. The Risk Estimator managed to detect all dangerous situations in $\mathcal{S}_{RE}$ but was prone to give false alarms in some test cases.

The Risk Estimator's performance in $\mathcal{S}_{RE+MN}$ is hard to evaluate because of the small number of test runs that resulted in a dangerous situation. Unfortunately, it is possible that characteristics of the simulation environment itself may have contributed to some results, since no explanation based on characteristics of the safety system is available in some cases. For example, the simulations take some time to start up properly which can affect the cases where one of the vehicles starts close to the intersection.

A variable parameter common to each test case is the **start distance** for V_H, which we denote d_1. It represents the *initial spatial separation* between the vehicles when compared to the fixed parameter d_0. Instead of plotting the measured entities against d_1, another entity is introduced to represent the *initial temporal separation*, Δt. Δt is defined as the *estimated difference in time between when V_L and V_H will each reach the intersection* based on their initial positions d_0 and d_1, and the presented speed model for "go" in Figure 3.3.

8.1   Request Time

Time to Grant (TTG) was calculated as the time difference between *when a vehicle started its first request round* and *when the request was fully granted*. We expect low TTG values for V_H (denoted $TTG(V_H)$) since it has the highest priority and thereby should not have to wait for $\langle GRANT \rangle$ from V_L. For V_L (denoted $TTG(V_L)$) we expect a shark-fin pattern in the TTG-Δt plot, with low values for the lowest Δt, a sudden jump in TTG when Δt is low enough for V_L to have to wait for V_H before crossing the intersection and then a steady decrease in TTG as V_L has to wait shorter and shorter times.

Recorded $TTG(V_L)$ and $TTG(V_H)$ for $\mathcal{ES}^{RE+MN}_{Noise}$ and $\mathcal{ES}^{RE+MN}_{Comloss}$ can be found in Figure 8.1. The added noise does not seem to affect TTG if compared to $\mathcal{TC}_{Normal}$ used for reference, denoted $N:0$, since the plotted points almost coincide for every value of Δt. $TTG(V_H)$ is close to 0 due to its empty membership, implying that it does not have to send out any requests. $TTG(V_L)$ starts higher than $TTG(V_H)$ since V_L reaches the intersection first when $\Delta t < 0$, and thus has to wait for a reply from V_H. A slight rise in $TTG(V_L)$ is visible after $\Delta t = -3$ and the plotted points then drop of linearly until they reach a value close to 0. This drop corresponds to V_H estimating that the time gap between them is now too small for V_L to safely perform its manoeuvre without V_H slowing down. V_L's requests are thereby rejected until V_H leaves the intersection. Increased Δt implies that V_H starts closer and closer to the intersection and V_L thereby has to wait for shorter and shorter time periods. A slight periodicity in $TTG(V_H)$ is distinguishable in the plot, with peaks at $\Delta t = -3, -1$, and 1. It is unclear what is causing these small periodic changes in TTG. Similar patterns are present in the communication loss plots to the right in Figure 8.1. Additional features in these plots includes an initially higher value for $TTG(V_L)$, temporary constant TTG instead of declining linear behaviour and initially declining linear behaviour. The first case occurs for CLD:0, CLP:2, the second case occurs for CLD:2, CLP:2 and CLD:0, CLP:2, and the third case for CLD:1, CLP:2. Case 1 and 3 are probably caused by communication loss during the request rounds. For example, in the third case V_L is trying to start a request round when its communication is switched off and it stays off for long enough for the vehicles to move close enough to produce a too small time gap.

A slight increase in $TTG\,(V_H)$ occurs after $\Delta t = 3$. This is an effect caused by V_H starting in a position when it has already passed the request line, $d_1 < d_{Init}$, and has to wait for the membership service to update its membership since a valid membership is needed to start a request round. This effect is thereby most probably caused by the slow upstart of the simulation environment and not a feature of the system. $\mathcal{S}_{MN}$ resulted in plots similar to $\mathcal{S}_{RE+MN}$ and the respective plots have been omitted. The results for TTG thereby live up to our expectations.
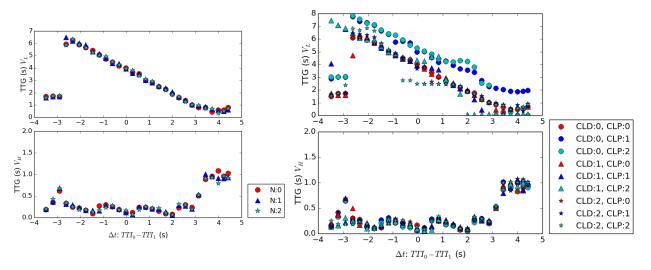

Figure 8.1: Time To Grant (TTG) for $\mathcal{ES}_{Noise}^{RE+MN}$ (left) and $\mathcal{ES}_{Comloss}^{RE+MN}$ (right). V stands for vehicle and N for noise level in the legend, where $N:0$ corresponds to the normal reference setup. $TTG\,(V_H)$– where V_H has higher priority – is fairly constant and keeps low while $TTG\,(V_L)$ approximates a sharkfin pattern. The sudden increase in TTG for both vehicles near $\Delta t = -3$ indicates a threshold value, after which V_H estimates that the gap between the two vehicles is too small and – as a result – it should not grant V_L.

8.2   Priority Violation Effects

Time Lost due to Priority Violation (TLPV) was estimated for V_H by taking the difference between the actual Time To Intersection (TTI) with the calculated ideal TTI when V_H follows the speed model with the intention "go". We expect values close to 0 for $\mathcal{S}_{MN}$ and $\mathcal{S}_{RE+MN}$ for the test cases $\mathcal{TC}_{Normal}$ and $\mathcal{TC}_{Noise}$ since the Manoeuvre Negotiation protocol should preserve the initial priorities. In $\mathcal{TC}_{Offender}$ and $\mathcal{TC}_{Comloss}$, however, we expect to see increased TLPV due to emergency break (EB) and V_L's lost messages. The measured TLPV for $\mathcal{ES}_{Noise}^{RE+MN}$ and $\mathcal{ES}_{Comloss}^{RE+MN}$ can be found in in Figure 8.2. The different noise levels tested do not seem to have a visible effect on TLPV when compared to the normal case, denoted **N:0**. The shape of the normal curve can also be distinguished in the result plot for the $\mathcal{TC}_{Comloss}$ when the results for varied **Communication Loss Period** (CLP) and the point of started communication loss, defined as **Communication Loss Distance** (CLD), are plotted together. Here we see a clear impact of both CLP and CLD on TLPV. The lower Δt values, corresponding to V_L starting closer to the intersection, are affected by $CLD:2$, which corresponds to communication loss for V_L just before entering the intersection. For these cases, V_H has to wait for V_L to start communicating again before its request can be granted. In the same way, CLD:1 gives rise to upgoing trends in the middle of the plot and CLD:0 a bit further right. The different slopes for the same CLD is caused by different CLP values where the highest CLP, corresponding to CLP:2, gives the steepest slope.

Figure 8.3 displays the measured TLPV for $\mathcal{ES}_{Offender}^{RE+MN}$. The TLPV has a peak just before $\Delta t = 0$ which is caused by V_H's RE triggering an EB due to V_L's unexpected behaviour.
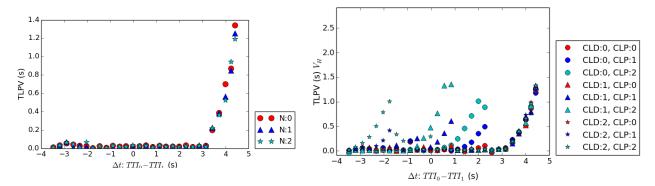
Figure 8.2: TLPV for $\mathcal{ES}_{Noise}^{RE+MN}$ (left) and $\mathcal{ES}_{Comloss}^{RE+MN}$ (right). In $\mathcal{ES}_{Noise}^{RE+MN}$ TLPV is kept below $0.1\,\mathrm{s}$ up for $\Delta t < 3$. A steady rise after $\Delta t = 3$ is present in both plots and is caused by a startup delay in the simulation. Additional rising trends for different CLD and CLP values are present in $\mathcal{ES}_{Comloss}^{RE+MN}$.
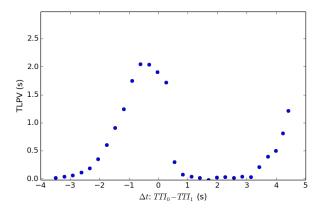

Figure 8.3: Time lost due to priority violation for V_H for $\mathcal{ES}_{Offender}^{RE+MN}$. Compared to the $\mathcal{ES}_{Noise}^{RE+MN}$ in the left plot in Figure 8.2 this plot has an additional peak just before $\Delta t = 0$, caused by EBs for V_H when V_L disobeys priorities and drives out in front of V_H.

The results of test cases in $\mathcal{S}_{RE}$ are almost identical to $\mathcal{S}_{RE+MN}$ and their plots are therefore omitted. There is one exception, $\mathcal{ES}_{Offender}^{MN}$, which is similar to $\mathcal{ES}_{Normal}^{RE+MN}$ since V_H is not equipped with a risk estimator (RE) in this test case.

The TLPV increases rapidly after about $\Delta t = 3$ for all setups and test cases. This is caused by the same issue as the one explained in Section 8.1, which causes V_H to wait for a valid membership. Waiting implies that the vehicle follows the "stop" speed model and therefore drives slower or even stops and waits just before the intersection causing a rise in TLPV. Apart from this, the results here matches our expectations with low TLPV in $\mathcal{TC}_{Normal}$ and $\mathcal{TC}_{Noise}$ and some higher, but still – for safety reasons – agreeable, values for $\mathcal{TC}_{Comloss}$ and $\mathcal{TC}_{Offender}$.

8.3 Emergency Break Events

An Emergency Break event happens if the Risk estimator detects a risk over a set threshold. We expect none or just a few emergency breaks in $\mathcal{S}_{RE+MN}$ since the vehicles actively negotiate their internal priority order and the Risk Estimator is notified of any priority change. In $\mathcal{S}_{RE}$ we expect the number of EB triggered by V_H to be higher than the number of EB triggered by V_L since V_H in this setup is expected to go throughout the simulation and thereby cannot have an expectation to stop and intention to go – which is what triggers

34

an EB.

The mean number of emergency break events for the $\mathcal{ES}_{Noise}^{RE}$ and $\mathcal{ES}_{Noise}^{RE+MN}$ scenarios is plotted in Figure 8.4. In $\mathcal{S}_{RE}$, increased noise seems to reduce the number of emergency breaks. This may be due to less clear intention caused by lack of new information, leading to a more evened out distribution of estimated intentions and expectations and less mismatch between the two. V_L shows a low number of emergency breaks which is expected since it is the lower priority vehicle and in an ideal case should not rise any alarms when the higher priority vehicle approaches the intersection. That emergency breaks are at all recorded for V_L may be caused by random functions used in the risk estimator and its gap model or due to randomness in the simulation.

In $\mathcal{ES}_{Noise}^{RE+MN}$, just a few emergency breaks were recorded and out of these no mean exceeded 1 for any Δt. This implies that on average less than 1 emergency break per run was recorded per measurement set of 10 runs. The reason why almost no emergency breaks were triggered is because the manoeuvre negotiation protocol used in $\mathcal{S}_{RE+MN}$ enables the vehicles to be aware of the other vehicle's intention and also informs the risk estimator of any priority change. In a best case scenario we would expect no emergency breaks but the few recorded is probably again due to randomness connected to the risk estimator or the simulation.
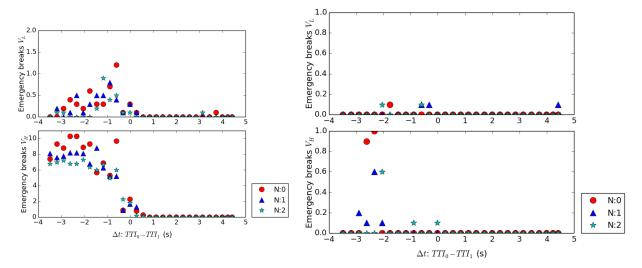

Figure 8.4: Mean number of emergency breaks for $\mathcal{TC}_{Normal}$ (N:0), and $\mathcal{TC}_{Noise}$ (N:x) for V_L and V_H seen in the legend as V:0 and V:1. The left plot is the result from $\mathcal{S}_{RE}$ and the right plot is from $\mathcal{S}_{RE+MN}$. Higher noise levels seem to lead to less number of emergency breaks. In $\mathcal{S}_{RE+MN}$ just a few emergency breaks were triggered. Notice that 0.1 implies 1 triggered emergency break in 10 runs.

In $\mathcal{ES}_{Comloss}^{RE}$ a drastic drop in emergency breaks by V_H can be seen when V_L stops communicating in the intersection, labelled as CLD:2 in the left plot in Figure 8.5. The largest drop is achieved for the longest communication loss period, denoted CLP:2. The result from $\mathcal{ES}_{Comloss}^{RE+MN}$ is displayed to the right in Figure 8.5. Here, we see an increase in the mean number of emergency breaks when compared to $\mathcal{ES}_{Normal}^{RE+MN}$ in Figure 8.4. However, most data points still lies beneath 1 and much lower than $\mathcal{S}_{RE}$.

8.4 TTCP Upon First Emergency Break

The **Time To Collision Point** (TTCP) for a vehicle is calculated when the first emergency break (EB) is triggered to give a measure of how far in advance a potential can be detected. We expect that if any EB are triggered then they will occur when the other vehicle reach a certain distance from the intersection since the vehicles strictly follow the speed models. This would result in a declining TTCP since it is measured for the vehicle that triggered the EB.
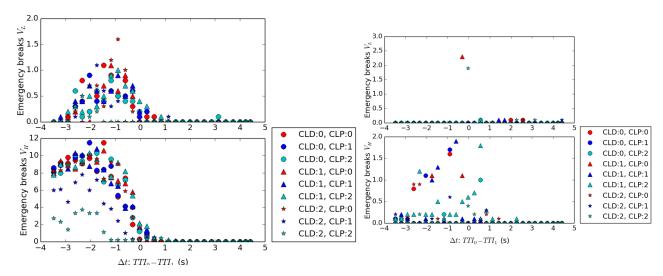
Figure 8.5: Mean number of emergency breaks for the $\mathcal{ES}\,^{RE}_{Comloss}$, left, and $\mathcal{ES}\,^{RE+MN}_{Comloss}$, right, for V_L and V_H for different communication loss periods (CLP) with varying start of the communication loss, CLD. The number of emergency breaks in $\mathcal{ES}\,^{RE}_{Comloss}$ is drastically reduced when V_L stops communicating in the intersection, corresponding to CLD:2. More emergency breaks were recorded in $\mathcal{ES}\,^{RE}_{Comloss}$ compared to $\mathcal{ES}\,^{RE}_{Normal}$, as seen in the left plot of Figure 8.4.

The results for $\mathcal{ES}\,^{RE}_{Normal}$, $\mathcal{ES}\,^{RE}_{Noise}$ and $\mathcal{ES}\,^{RE}_{Comloss}$ is plotted in Figure 8.6. All three test cases show a similar linear decline of about -1 for V_H when $\Delta t < 0$ which indicates that the EB is triggered when V_L is at approximately the same distance from the intersection in each case, approximately 1 s away from reaching the intersection. A decrease in TTCP is visible for $\mathcal{TC}_{Comloss}$ for increasing communication loss time (CLP) for V_H with $CLD : 2$, corresponding to communication loss for V_L when entering the intersection. Both plots show a few false alarms for $\Delta t > 0$ for V_L, and a lot of late alarms (TTCP< 0) for $\Delta t < 0$. $\mathcal{TC}_{Offender}$ is similar to $\mathcal{TC}_{Normal}$ and $\mathcal{S}_{RE+MN}$ is similar to $\mathcal{S}_{RE}$ but with less data points, so these plots are omitted. The linear decline of TTCP for V_H matches our expectation that the detection would occur when V_H reaches a certain distance from the intersection. The late EB triggered by V_L was not expected and may have been caused by unpredictability or implementation flaw of the risk estimator.

8.5 Collisions and Dangerous Situations

Collisions between the two vehicles was calculated as when both vehicles at the same time were inside the area common to their trajectories if considering the vehicle width. A dangerous situation occurred if their fronts where closer than 4 m apart, which implies that collisions are a subset of dangerous situations. We expect to see no collisions and at most a few dangerous situations in $\mathcal{S}_{RE+MN}$ and $\mathcal{S}_{MN}$ except for in $\mathcal{TC}_{Offender}$ where V_H may not have enough time to detect and stop for the offender (in $\mathcal{S}_{RE+MN}$). Collisions may on the other hand be present in $\mathcal{S}_{RE}$ since the Risk Estimator has a best effort approach to safety.

The number of recorded collisions and dangerous situations for different Evaluation Scenarios can be found in Table 8.1. $\mathcal{S}_{MN}$ and $\mathcal{S}_{RE+MN}$ yielded lower values than $\mathcal{S}_{RE}$ in all cases. $\mathcal{S}_{MN}$ resulted in less collisions than $\mathcal{S}_{RE+MN}$ while $\mathcal{S}_{RE+MN}$ resulted in less dangerous situations than $\mathcal{S}_{MN}$. The highest number of collisions and dangerous situations for $\mathcal{S}_{RE}$ seems to occur when a communication loss occurs just before V_L reaches the request line.

The collisions in the $\mathcal{S}_{RE+MN}$ scenario occurred in $\mathcal{ES}\,^{RE+MN}_{Offender}$, which was expected, but also in $\mathcal{ES}\,^{RE+MN}_{Comloss}$, which was not expected. Either, the longest period of communication loss for V_L in $\mathcal{ES}\,^{RE+MN}_{Comloss}$ prevented both the Risk Estimator and Manoeuvre Negotiation protocol from keeping the vehicle safe in this one case, or this collision was caused by an undetected error in during the simulations. $\mathcal{S}_{RE}$ resulted in a significantly
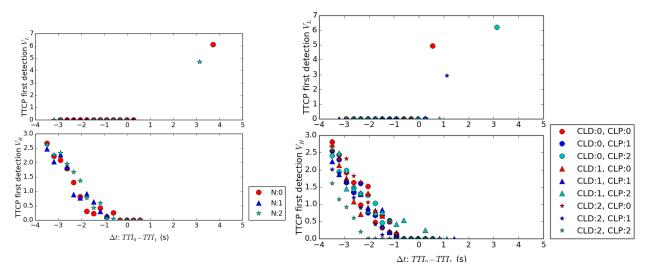
Figure 8.6: Time To Collision Point (TTCP) for the earliest EB for $\mathcal{ES}^{RE}_{Normal}$ and $\mathcal{ES}^{RE}_{Noise}$ (left) and $\mathcal{ES}^{RE}_{Comloss}$ (right). Both figures show a linearly declining TTCP with increasing $\Delta t < 0$ for V_H and a few false alarms (where $\Delta t > 0$).

higher amount of collisions and dangerous situations as expected.

	Collision-RE	Collision-MN	Collision-RE+MN	Danger-RE	Danger-MN	Danger-RE+MN
Normal	5	0	0	9	0	0
N0	5	0	0	8	1	0
N1	1	0	0	8	0	0
O	2	1	2	4	4	3
CL0-0	5	0	0	8	1	0
CL0-1	3	0	0	9	1	0
CL0-2	4	0	0	8	1	0
CL1-0	2	0	0	9	0	0
CL1-1	4	0	0	10	0	0
CL1-2	7	0	0	9	2	0
CL2-0	3	0	0	7	2	0
CL2-1	4	0	0	9	0	0
CL2-2	2	0	1	8	1	1

Table 8.1: Number of test cases that resulted in a collision or a dangerous situation in $\mathcal{S}_{RE}$, $\mathcal{S}_{MN}$, and $\mathcal{S}_{RE+MN}$ for the $\mathcal{TC}_{Normal}$ (N0), $\mathcal{TC}_{Noise}$ (Nx), $\mathcal{TC}_{Offender}$ (O), and $\mathcal{TC}_{Comloss}$ (CLX-Y). The maximum value is 29 corresponding to at least one collision or dangerous situation in all cases with different start distance d_1. $\mathcal{S}_{MN}$ and $\mathcal{S}_{RE+MN}$ resulted in much less collisions and dangerous situations than $\mathcal{S}_{RE}$.

8.6 Risk Estimator Recall and Precision

For the Risk Estimator's Emergency Breaks, recall was evaluated as total number of tests where an EB was triggered during a dangerous situation divided by the total number of cases where a dangerous situation occurred. Precision was calculated as the number of true dangerous situations that were detected and triggered an EB divided by the total number of cases were an EB was triggered. We expect a high Recall

since the vehicles follow the same motion patterns as the Risk Estimator uses to infer their manoeuvre intention.

Recall and precision values for $\mathcal{S}_{RE}$ and $\mathcal{S}_{RE+MN}$ are presented in Table 8.2. V_H has a recall value of 1.0 in $\mathcal{S}_{RE}$ for all test cases, which implies that it was able to detect all dangerous situations despite increased noise, communication loss, and the other vehicle ignoring priorities. However, the lower precision results implies that the Risk Estimator is prone to rise false alarms. Safety is prioritised over false alarms by the Risk Estimator when run by itself. The result for $\mathcal{S}_{RE+MN}$ is less clear since only a few dangerous situations were recorded for this setup, see Table 8.1.

In $\mathcal{S}_{RE}$ we got high recall values as expected while more tests will have to be run in order to get a more reliable values for $\mathcal{S}_{RE+MN}$.

	Recall-RE	Recall-RE+MN	Precision-RE	Precision-RE+MN
Normal	0.56/1.0	-/-	0.28/0.50	-/-
N0	0.44/1.0	-/-	0.22/0.50	0.0/0.0
N1	0.38/1.0	-/-	0.18/0.47	-/-
O	1.0/1.0	0.67/1.0	0.19/0.19	0.10/0.15
CL0-0	1.0/1.0	-/-	0.4/0.4	-/-
CL0-1	1.0/1.0	-/-	0.38/0.38	-/-
CL0-2	0.75/1.0	-/-	0.29/0.38	0.0/0.0
CL1-0	0.78/1.0	-/-	0.32/0.41	-/-
CL1-1	0.50/1.0	-/-	0.24/0.48	0.0/0.0
CL1-2	0.89/1.0	-/-	0.33/0.38	0.0/0.0
CL2-0	1.0/1.0	-/-	0.33/0.33	0.0/0.0
CL2-1	0.44/1.0	-/-	0.21/0.47	0.0/0.0
CL2-2	0.25/1.0	1.0/0.0	0.12/0.47	0.5/0.0

Table 8.2: Recall and precision results for $\mathcal{S}_{RE}$ and $\mathcal{S}_{RE+MN}$ where values separated with a "/" are the values for V_L and V_H respectively. A "-" signifies that no value could be computed due to a lack of dangerous situations or emergency breaks. The Risk Estimator performs better on Recall compared to Precision and V_H detects all dangerous situations in $\mathcal{S}_{RE}$.

9 Discussion

9.1 Preservation of the Original Traffic Rules

Traffic rules and conventions are designed not only to make driving more safe but also to promote fairness and a high traffic flow. It is therefore important that our combined safety system tries to adhere to the traffic rules and conventions.

It is worth noting that a vehicle p_i following the protocol will only grant another vehicle p_j if p_i predicts that p_j will not interfere with p_i's intended manoeuvre. This is an egoistic approach, since p_i is not prepared to slow down for a lower priority vehicle to, for example, increase the overall throughput. Further more, p_j will only ask p_i if p_i is in p_j's membership – p_i has higher priority than p_j, where lower priority reflects which vehicle that according to the traffic rules should give-way. The combination of the egoistic approach to answering a request and the composition of the memberships therefore implies that the protocol in a failure free traffic situation should preserve priorities set by the traffic rules. This is also shown in the simulations, Figure 8.2, where the priority vehicle is shown to slow down for the lower priority vehicle only if communication is lost during the request rounds.
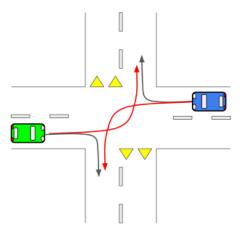
Figure 9.1: An example of conflicting manoeuvres in red – both vehicles turning left – and non-conflicting trajectories in black – both vehicles turning right.

9.2 Suggested extensions

The Manoeuvre Negotiation Protocol was extended during this project and more extensions are possible. Some of the ones we have identified includes

- **Allowing multiple granted vehicles**. The presented version of the protocol only allows a vehicle to grant one other vehicle at the time. A possible way of extending the protocol to allow multiple held grants without reducing safety is to divide the intersection into multiple critical sections instead of just one by defining conflicting manoeuvres, see Figure 9.1. An agent receiving a request can thereby base its decision to grant or deny on if the intended manoeuvres overlap in time *and* space. For example, two vehicles coming from opposite directions that both intend on doing a left turn through an intersection will have conflicting manoeuvres, while if they both intend on doing a right turn then their manoeuvres are not conflicting.

- **Extending to more traffic situations**. So far we have only tested the protocol on intersections and it would be interesting to see how well it performs in other traffic situations. It is possible to extend the priority matrix (Section 3.2.1) to be valid for roundabouts, merging lanes and possibly general roads to indicate how the traffic priorities are set up and how the memberships for the manoeuvre negotiation should be composed. An extension would also have to be made to the speed models to capture normal driving patterns in these situations.

- **Periodically changing the default priorities for fairness and deadlock avoidance**. The default priorities are static in our simulations to reflect the rules in an unsignalled intersection. An extention to signalled intersections could potentially be simulated by letting the default priorities change, indicating a change of priority road. This change would only have to be made by the membership service and would be made present to the vehicles by a changed membership. Changing the default priorities is also a way of increasing fairness and avoiding deadlocks for lower priority vehicles when there is a continuous flow of vehicles on the priority road.

9.3 Suggested further testing

- **Varied turns and intersections.** There are more turn combinations than going straight and turning left that may cause collisions in an intersection. It would therefore be beneficial to run more tests with different combinations of manoeuvre intentions. There are also many more intersection types than

just 4-way give-way intersections which also could be constructed in our simulation environment and tested.

- **Recorded real driving data.** Using real driving data instead of relying on speed models and fixed paths could give a better indication on our system's performance in real traffic scenarios. Recorded data from real vehicles in a 4-way, give-way intersection would then be needed to conduct the tests. The PID-controller on every vehicle in the simulation could be set to follow the recorded path and the recorded speed could function as a speed model.

- **Network simulator.** A network simulator could be used to simulate packet delays and message loss in a more realistic way. Packets sent in our setup in ROS are delivered instantly without any delay which would never happen in a real traffic scenario.

- **Multiple vehicles.** It could also be worth testing the system's performance when more then 2 vehicles are approaching the intersection, especially if a multi-grant version of the protocol is developed. Adding more vehicles to the simulation would make the intersection more busy and it would be interesting to evaluate the safety and throughput in this scenario as well.

9.4 Deadlock avoidance

Deadlocks are possible in this system since the four conditions for causing deadlocks: mutual exclusion, resource holding, no preemption, and circular wait [5] could be present in the system if the algorithm is not designed properly. Mutual exclusion is used for the whole (or in possible extensions: parts of) the intersection. Resource holding may occur since vehicles requesting for a permission may be required to get a grant from more than one vehicle and vehicles are only allowed to grant one vehicle at the time. No preemption is present if explicit release (section 3.1) is used since the grant is held until a release message is received or the granted vehicle has left the intersection. Circular wait is possible if more than one vehicle are sending their requests concurrently since vehicles are only allowed to grant one vehicle at any time and grants from all expected respondents are required to get permission to perform the manoeuvre. As an example of a deadlock situation: Lets call two agents that concurrently send requests $p_i and p_j$. If p_i's and p_j's sets of requires responses have at least two agents in common, say p_a and p_b then p_i may get a grant from p_a while p_j gets a grant from p_b due to message delays. Agent p_a then have to deny p_j since only one grant can be given at any time, and the same applies to p_b's response to p_i's request. If p_i's request is newer then p_i will release its request and grant p_j instead. But p_i will also release its request since it will receive a deny from p_b and the whole process starts over again.

Mutual exclusion for the intersection is vital for avoiding collisions so efforts should not be put there to remove that deadlock conditions. Getting a grant from every vehicle expected to answer is also a key feature to ensure safety, but it could potentially be possible to grant multiple vehicles and still avoid collisions. However, without multiple grants, what could be avoided is circular wait. If the vehicle with the newest request can be made to back off for a time long enough $(T_A + 2T_D)$ for the vehicle with the newest request to get a grant then circular wait will not be possible anymore.

Intersections also have pre-defined priorities which can have an impact on formations of deadlocks. Using time stamps to set priorities of requests enables the avoidance of these deadlocks if the function noPriorityViolationDynamic takes the pre-defined rules into account. A vehicle with a non-empty membership and that has not been fully granted should have the intention to stop and is not allowed to enter the intersection. This vehicle should therefore release its current request if it receives another request with an earlier time stamp.

10 Conclusion

In this thesis, a Manoeuvre Negotiation protocol intended for increasing the safety for autonomous vehicles have been presented. A correctness proof was supplied where the requirements set on the protocol was shown

to be fulfilled. The requirements stated limitation on the completeness of memberships, when manoeuvre negotiating requests can be initiated, how agents should answer to requests, when a grant should be released, and when an agent is allowed to enter the intersection.

Our simulations further show that the setups with the Manoeuvre Negotiation protocol (MN) used with or without the Risk Estimator (RE) are able to avoid collisions in more cases than when using merely RE. A higher priority vehicle equipped with a RE was shown to be able to detect all risky situation even in the presence additional noise, and communication loss. However, the detection of the risky situation was in many cases not made early enough for the vehicle to be able to break to avoid a collision. The throughput for the higher priority vehicle when MN was activated was further shown to just marginally deviate from the ideal case which also implies that the protocol preserves the priorities set by the traffic rules. Our combined system thereby states an example of how vehicle-to-vehicle communication can be used alongside existing safety systems to increase safety for autonomous vehicles.

Finally, possibilities in ways to extend to the protocol and additional testing methods were provided. Possible extensions included allowing vehicles to concurrently grant more than one vehicle at the time and broadening the scope in which the protocol can be used. Further testing is recommended to focus more on varying the driving patterns of the vehicle or even use reordered data from real vehicles to run the vehicles in the simulations.

References

[1] Matthias Althoff and Silvia Magdici. Set-based prediction of traffic participants on arbitrary road networks. *IEEE Transactions on Intelligent Vehicles*, 1(2):187–202, 2016.

[2] G. S. Aoude, V. R. Desaraju, L. H. Stephens, and J. P. How. Driver behavior classification at intersections and validation on large naturalistic data set. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):724–736, 2012.

[3] Mattias Brännström, Erik Coelingh, and Jonas Sjöberg. Model-based threat assessment for avoiding arbitrary vehicle collisions. *IEEE Transactions on Intelligent Transportation Systems*, 11(3):658–669, Sept 2010.

[4] António Casimiro and Elad M. Schiller. Membership-based Maneuver Negotiation in Safety-critical Vehicular Systems. Technical report, Chalmers University of Technology, Department of Computer Science and Engineering, 10 2017.

[5] Ed Coffman, M J. Elphick, and Arie Shoshani. System deadlocks. *ACM Computing Surveys*, 3(2):67–78, June 1971.

[6] Emelie Ekenstedt. Membership-based manoeuvre negotiation in autonomous and safety-critical vehicular systems. Master's thesis, Department of Computer Science and Engineering, Chalmers University of Technology, June, 11 2019.

[7] Xinli Geng, Huawei Liang, Biao Yu, Pan Zhao, Liuwei He, and Rulin Huang. A scenario-adaptive driving behavior prediction approach to urban autonomous driving. *Applied Sciences*, 7(4):426, 2017.

[8] F. Gustafsson. Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 25(7):53–82, July 2010.

[9] Christos Katrakazas. A new methodology for collision risk assessment of autonomous vehicles. In *Transportation Research Board 96th Annual Meeting*, 2017.

[10] M. Koschi and M. Althoff. Interaction-aware occupancy prediction of road vehicles. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, Oct 2017.

[11] Stéphanie Lefèvre, Christian Laugier, and Javier Ibañez-Guzmán. Intention-Aware Risk Estimation for General Traffic Situations, and Application to Intersection Safety. Research Report RR-8379, INRIA, October 2013.

[12] M. Liebner, F. Klanner, M. Baumann, C. Ruhhammer, and C. Stiller. Velocity-based driver intent inference at urban intersections in the presence of preceding vehicles, 2013.

[13] S. Noh and K. An. Risk assessment for automatic lane change maneuvers on highways. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 247–254, May 2017.

[14] F. Seeliger, G. Weidl, D. Petrich, F. Naujoks, G. Breuel, A. Neukum, and K. Dietmayer. Advisory warnings based on cooperative perception. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 246–252, June 2014.

[15] Ozan K Tonguz. Red light, green light—no light: Tomorrow's communicative cars could take turns at intersections. *IEEE Spectrum*, 55(10):24–29, 2018.

[16] J. Ward, G. Agamennoni, S. Worrall, and E. Nebot. Vehicle collision probability calculation for general traffic scenarios under uncertainty. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 986–992, June 2014.

[17] Guotao Xie, Xinyu Zhang, Hongbo Gao, Lijun Qian, Jianqiang Wang, and Umit Ozguner. Situational assessments based on uncertainty-risk awareness in complex traffic scenarios. *Sustainability*, 9(9), 2017.

[18] Mario Zanon, Sébastien Gros, Henk Wymeersch, and Paolo Falcone. An asynchronous algorithm for optimal vehicle coordination at traffic intersections this work was supported by copplar (project number 32226302), the swedish research council (vr, grant number 2012-4038) and the european commission seventh framework (adaptive, grant number 610428). *IFAC-PapersOnLine*, 50(1):12008–12014, 2017.

[19] A. Zyner, S. Worrall, J. Ward, and E. Nebot. Long short term memory for driver intent prediction. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1484–1489, June 2017.

[20] Alex Zyner, Stewart Worrall, and Eduardo Nebot. A recurrent neural network solution for predicting driver intention at unsignalized intersections. *IEEE Robotics and Automation Letters*, 3(3):1759–1764, 2018.

A An alternative implementation of noPriorityViolationStrict

The purpose of noPriorityViolationStrict is to determine if a priority violation can possibly occur between two vehicles when priorities are static. To draw the right conclusion, the algorithm has to consider both normal and extreme driving behaviour, which can be done with the help of occupancy prediction [1]. The main idea of occupancy prediction is to, given a start state of an agent, provide an upper limit to the agent's position some time later using constraints on vehicle motion.

Pseudocode for noPriorityViolationStrict is given in 4. The procedure takes four input parameters including the agent states of two vehicles, p_i and p_j, the time from which calculations should begin, t, and a time step δ_t. The states p_i and p_j are used to compute the initial occupancy area $\mathcal{O}_i^0$ and $\mathcal{O}_j^0$ at the time t.

An occupancy area $\mathcal{O}_i^k$ covers every possible position the agent represented by p_i can have in the time interval for which it was computed. Both physical and legal constraints, such as limited engine power and speed limits [1], can be applied to predict $\mathcal{O}_i^k$ from $\mathcal{O}_i^{k-1}$. The pseudocode uses the interface *getOccupancyArea* to calculate these areas.

The implementation approach for noPriorityViolationStrict considers the intersection of the occupancy areas $\mathcal{O}_i^k \cap \mathcal{O}_j^k$ for consecutive time intervals $\Delta t_k = [t + k\delta_t, t + (k+1)\delta_t]$, where $k = 0, 1, 2....$ Traffic priorities are preserved by letting an agent p_k grant a request from another agent p_j if, and only if, $\mathcal{O}_i^k \cap \mathcal{O}_j^k = \emptyset \forall k \in [0, M] : ((hasLeftIntersection(\mathcal{O}_i^M) \vee hasLeftIntersection(\mathcal{O}_j^M)) \wedge (\neg hasLeftIntersection(\mathcal{O}_i^{M-1}) \wedge$

$\neg hasLeftIntersection(\mathcal{O}_j^{M-1})$), that is, Δt_M is the first time interval after which p_i or p_j is guaranteed to have left the intersection.

Further constraints in the predicted vehicle motions can be applied if multiple vehicles are present in a traffic scene [10]. Let $\{p_i\}_{i=0^N}$ be N vehicles in a queue on a one-lane road where p_1 is the leading vehicle. The occupation areas $A_j(p_i)\forall i \in [2, M]$ will then be bounded in the forward direction by $A_j(p_1)$ at any time instant j as long as the road does not expand to a multi-lane road.

The presented approach should be able to predict all possible priority violations caused by *predictable* vehicle motion since the approach relies on the physical constraints of the vehicle. We define predictable vehicle motion as all kinds of driving in the absence of rare natural events such as appearing sink holes, rockfall, sudden flooding etc, which can cause the vehicle to move in unexpected ways.

This version of noPriorityViolation was selected to be used by the membership service to ensure that a membership for an agent p_i contains all other vehicles that could potentially cause a priority violation with p_i. However, this approach is not efficient regarding throughput in normal traffic situations where most vehicles follow normal trajectory and speed patterns. We therefore decided to employ another version of noPriorityViolation based on behaviour models and intersection occupancy intervals for making the grant or deny decision. This version is presented in Section 3.2.5.

Algorithm 4: noPriorityViolationStrict relying on static priorities.

---

1 **Structures:**
2 $AgentState = (ta, x, v, a);$ `// timestamp, position, velocity, acceleration`
3
4 **Interfaces:**
5 $getOccupancyArea(t_s, t_f, p)$: ;
6 $hasLeftIntersection(\mathcal{O}(p))$: returns True if all predicted states of p, $\mathcal{O}(p)$, are outside and moving away from the intersection;
7
8 **procedure** noPriorityViolationStrict(p_i, p_j, t, δ_t) **begin**
9   **let** $t_c = t;$
10   **let** $\mathcal{O}_i = getOccupancyArea(p_i.ts, t, p_i);$
11   **let** $\mathcal{O}_j = getOccupancyArea(p_j.ts, t, p_j);$
12   **while** $\neg(hasLeftIntersection(\mathcal{O}_i) \vee hasLeftIntersection(\mathcal{O}_j))$ **do**
13   **let** $\mathcal{O}_i = getOccupancyArea(t_c, t_c + \delta_t, \mathcal{O}_i);$
14   **let** $\mathcal{O}_j = getOccupancyArea(t_c, t_c + \delta_t, \mathcal{O}_j);$
15   **if** $\mathcal{O}_i \cap \mathcal{O}_j \neq \emptyset$ **then return** False;
16   **let** $t_c = t_c + \delta_t;$
17   **return** True;