Collaborative Driving System Using Teamwork for Platoon Formations

Simon Hallé
Département d'informatique et génie logiciel
Université Laval
Sainte-Foy, QC, Canada, G1K 7P4
halle@iad.ift.ulaval.ca

Brahim Chaib-draa
Département d'informatique et génie logiciel
Université Laval
Sainte-Foy, QC, Canada, G1K 7P4
chaib@iad.ift.ulaval.ca

Abstract

Collaborative driving is a growing domain of Intelligent Transportation Systems (ITS) that makes use of communications to autonomously guide cooperative vehicles on an Automated Highway System (AHS). In this paper, we address this issue by using a platoon of cars considered as more or less autonomous software agents. To do that, we propose a hierarchical architecture based on three layers (guidance layer, management layer and traffic control layer) which can be used to develop centralized platoons (where a head vehicle-agent coordinates other vehicle-agents by applying its coordination rule) and decentralized platoons (where the platoon is considered as a team of vehicle-agents trying to maintain the platoon). The latter decentralized model will mainly consider a teamwork related model using architectures like STEAM. These different coordination models will be compared using simulation scenarios to provide arguments for and against each approach.

1. Introduction

Transport systems all over the world are suffering from spreading problems regarding mainly their traffic flow and safety. To address these traffic problems, we generally build more highways, but this solution is greatly limited by the available land areas, which is running low in most cosmopolitan cities. An alternative solution which is growing in popularity is to develop techniques that increase existing roads' capacity by investing in Intelligent Transportation Systems (ITS) infrastructure [10]. It is shown that ITS may provide potential capacity improvements as high as 20 percent [25]. ITS can be seen as a complex set of technologies that are derived from information and computer technologies, to be applied to transport infrastructure and vehicles [16]. We can cite as ITS components: advanced transportation management, advanced transportation information system, and commercial vehicle operations. Among these components, there are sub-components such as automobile collision avoidance and electronic guidance system. These sub-components are generally sustained by individual technologies as: electronic sensors, wire and wireless communications, computer software and hardware, GPS, GIS, etc. The main objectives of ITS include: reduce environmental impacts, enhance safety, reduce congestion, etc.

Collaborative driving is an important subcomponent of ITS that strives to create vehicles being able to cooperate in order to navigate through highway traffic using communications. Such a system is made possible with the collaboration of a lower layer of control system, which acts as an Adaptive Cruise Control (ACC) [12]. Thus, at its simplest implementation, collaborative driving will add a layer of communication to the present ACC, to create a Cooperative Adaptive Cruise Control (CACC) and benefit from a communication system to collaborate between vehicles' ACC. The addition of such a communication system has proved to be a very positive addition to ITS [31], by helping collaboration from one vehicle to another. Going forward to a wider level of collaboration, the vehicle platoon model, has used communications to coordinate platoon members with their platoon leader [29]. Compared with CACC, this vehicle organisation adds a deliberative system in the lead vehicle, which will coordinate the preceding vehicles equipped with CACC, to maintain the platoon formation. As a new approach to this centralized coordination system, we aim to incorporate the Multiagent vision to the platoon architecture and coordinate the vehicles through teamwork for agents models [27]. Such an approach in-

corporates autonomous agents in each vehicle that make use of the communication system, to coordinate each others in a decentralized platoon model.

In this paper, we address the coordination issue for a platoon of vehicles, by first describing the collaborative driving domain and the simulator used to represent this environment, in section 2. Then, section 3 presents the hierarchical architecture we adopted as the driving system of automated vehicles. Section 4 describes the different coordination strategies we implemented and tested in the previous simulator. Section 5 reports the preliminary results using the comparison of centralize coordination approaches, to decentralized ones, using teamwork. Finally, section 6 presents a discussion, followed by the conclusion.

2. Domain of application

Collaborative driving is a research domain which aims to create automated vehicles that collaborate in order to navigate through traffic. In this sort of driving, one generally form *a platoon* [30], that is a group of vehicles whose actions on the road are coordinated by the means of communication. The first vehicle of a platoon is called the platoon leader and its role is to manage the platoon and guide it on the road. Our work comes within this framework and is a part of the Auto21 project [4][5], a member of the Canadian Networks of Centres of Excellence, studying the automobile of the 21st century within three levels of system functionality [9], examined in parallel.

- *In the **first** level* (autonomous longitudinal control), only the relative distance and velocity of the cars will be actively controlled in a type of generalized and distributed "cruise control system" (or ACC). At this level, steering will be controlled by the human driver, and the platoon's leader will be driven by an expert human driver without any autonomous control system.
- *In the **second** level* of complexity (semi-autonomous longitudinal-lateral control), the relative lateral and longitudinal motion of each vehicle, considering the vehicle preceding it, will be autonomously controlled all the way up to the platoon's "lead car", in a form of generalized car-train.
- *In the **third** level* (fully autonomous longitudinal-lateral control), the addition of cooperative steering, using the road and the telematic infrastructure as a guide for absolute motion control, will provide autonomous road-following capabilities.

2.1. Collaborative Driving Simulator

The environment in which our vehicle coordination system has been tested is a Collaborative Driving System (CDS) [8] simulator developed to provide user interface and graphical results of our work. Similar to traffic simulators like Carnegie Mellon's SHIVA [26], or California Path's Smart AHS [3], our simulator called HESTIA (a screen shot of it, is presented in Figure 1), aspires to a lower level of vehicle simulation, as its main purpose is to create an environment for the development and testing of Intelligent Transport Systems (ITS). To do so, it simulates a highway environment, with vehicles represented as 3D shapes, which are using simulated dynamics and sensors to retrieve information from the environment, such as the vehicle's internal dynamic information and external vehicles' dynamic information. The simulator's environment is based on JAVA 3D^{TM}'s technology, which offers a 3D environment in which autonomous vehicles can evolve.

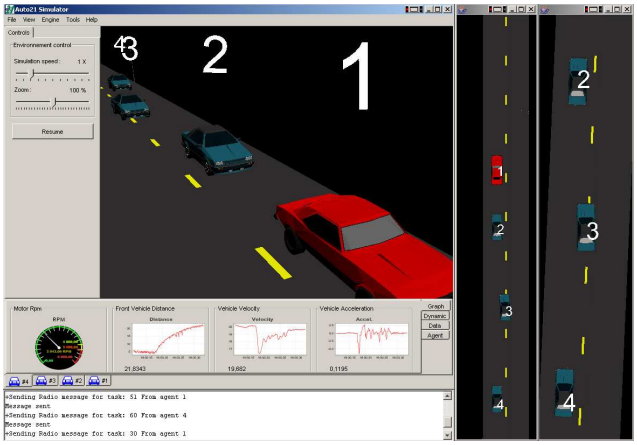

Figure 1. The merge within a platoon formation in HESTIA 3D simulator.

The simulated vehicles' model includes longitudinal and lateral vehicle dynamics, wheel model dynamics, engine dynamics, torque converter model, automatic gear shifting and throttle/brake actuators. The engine and transmission torque converter and differential was translated from a model developed under MATLAB/SIMULINK by our partners at Sherbrooke University [11]. The wheel model and vehicle's lateral and longitudinal dynamics were developed using the theory on wheel slip, tyre side slip angle and friction co-efficients applied to a single-track model, as well as the theory on the chassis' motions models, described

in [14]. The simulated sensors were developed using the 3D engine of JAVA 3D^{TM}, and for the current test, each following vehicle are equipped with a vehicle-based laser sensor for a low-level, inter-vehicle navigation. This sensor provides information on the front object's (a vehicle) distance and difference of velocity, for distances up to 100 m, using an abstract model of laser. The second type of sensor, used for high-level navigation, is a Global Positioning System (GPS), which gives real-time information on the vehicle's position (latitude, longitude), mapped in a two dimensions system. Finally, we simulated a radio transmitter/receiver onboard each vehicle for two ways point to multipoint communications. This communication model includes adjustable delay, throughput and protocol for different communication devices. We will not go further on a detailed representation of the simulator's components, as it is out of scope for this paper.

Within the hierarchical architecture presented in section 3, a driving agent is able to use the simulated sensors, actuators and communication system to interact with the Automated Highway System. The overall representation of the simulated models and its user are show in Figure 2.

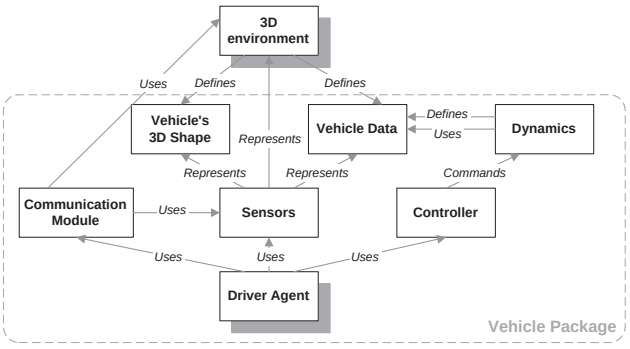

Figure 2. Vehicle simulation environment.

2.2. Simulated driving scenarios

The main scenario of our studies is the maintenance of the platoon formation, so the two scenarios we will focus on will be the two main disturbance in this formation: a vehicle splitting, a vehicle merging the platoon. Those two scenarios, represented in Figure 3, can be detailed as follows for a better understanding:

A Vehicle splitting happens when a vehicle member of a platoon decides to leave it, thereby forming two non-empty platoons. To execute this manoeuvre, the splitter (*F2* in Figure 3) must communicate its intention of leaving the platoon, so the platoon formation modifies the distances at the front and rear of the splitting vehicle as shown in *step 1* (*S1*) of Figure 3. When this new formation gains stability, the splitting vehicle *F2* can change lane, while the rest of the platoon followers keep the same distances. When the splitting vehicle safely left the platoon (*S2*), the gap created for its departure can be closed, thus forming back the precedent platoon, minus one vehicle (*S3*).

A Vehicle merging is the exact opposite of a split manoeuvre: two non-empty platoons merge together to become one. This manoeuvre requires a platoon formed of only one vehicle, which is *L2* in Figure 3, to communicate to another platoon its will to join it. Moving from *S1* to *S2*, the latter platoon will react by creating a safe space and communicating to the merging vehicle the dynamic position of this space in its platoon. The merging vehicle modifies its velocity to join the meeting point, verifies if it is safe to merge and changes lane to enter the platoon formation and leave *S2* to go to *S3*. Once the merged vehicle has stabilized its inter-vehicle distance, the platoon can reach its precedent formation plus one vehicle, by diminishing the distances with the new vehicle. Although, the steps of the merge task may differ from one coordination approach to another, this represents the general pattern of the merge manoeuvre.

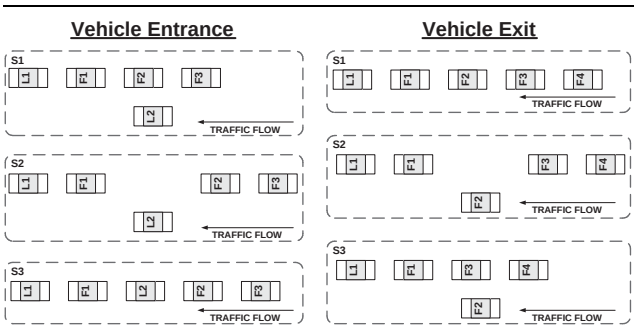

Figure 3. The three steps of the removal (split) and insertion (merge) of a vehicle in the platoon.

As it has been shown, the insertion and removal of a vehicle are the most problematic cases of the platoon formation and this is why the different coordination models presented in section 4 will focus on the communications involved during those tasks. Since we focus on the communication and coordination of the platoon, in this first level of complexity, the lateral auto-

mated control will be simulated to enable us to perform vehicle entrance and exit from the platoon. The lateral guidance system of our current system could then be seen as the simulation of the human driver's steering behavior, or a first phase of the lateral guidance system. This subject being out of scope from this paper which focuses on communications, we will not detail the lateral controller.

3. Hierarchical Architecture for Collaborative Driving

The architecture we adopted for our driving system is based on a hierarchical approach [4]. This model uses a more reactive system as the bottom of the architecture and moves forward to a more deliberative system as it raises to the upper levels. This approach was inspired by widely used hybrid behavioral architectures [22], but organized in a more hierarchical manner [2]. This model could be compared to precedent agent oriented architectures, also organized in a hierarchical way [7]. The chosen architecture also adds a coordinating system as the top of this architecture which interacts mainly with the precedent upper deliberative module, also inspired from other successful architectures [13]. Finally, as we related to other collaborative driving models, our hierarchical architecture was also inspired by Tsugawa's architecture [28] and other concepts coming mainly from the PATH project [17]. The resulting architecture has three major layers: *guidance layer*, *management layer* and *traffic control layer*, as indicated in Figure 4.

The *guidance layer* has the function of sensing the conditions and states ahead and around the vehicle and activating the longitudinal or the lateral actuators. For the sensing systems, inputs come from sensors for speed, acceleration, raw rate, machine vision, etc. This layer also outputs sensing data and vehicles state variables to the vehicle guidance layer and then receives steering and vehicle velocity commands from the same guidance layer. These considerations have lead us to divide this layer in *intelligent sensing* and *vehicle control* sub-layers as depicted in Figure 4. The vehicle control will be taken care of, by our collaborators at Sherbrooke University [11].

The *management layer* determines the movement of each vehicle under the cooperative driving constraints using data from (a) the guidance layer, (b) vehicles coordination constraints through the inter-vehicle communication, (c) the traffic control layer through the road-vehicle communication. To determine the movement of each vehicle under the cooperative constraints, this layer needs to reason on the place of the vehicle in the platoon when this platoon remains the same (intra-platoon coordination), and its place in a new platoon when this platoon changes (inter-platoons coordination). The first type of coordination is handled by the *networking* module and the second by the *linking* module, together forming the *coordination* sub-layer. Generally, the task of the *linking* module is to communicate with the traffic control layer to receive suggestions on actions to perform. Resulting from these suggestions, the agent's *linking* module will try to coordinate inter-platoon actions like: join, split and lane-change. This layer is also responsible of maintaining a safe inter-platoon distance which will also define a desired velocity and inter-vehicle spacing for platoon members. This intra-platoon policy will be maintained

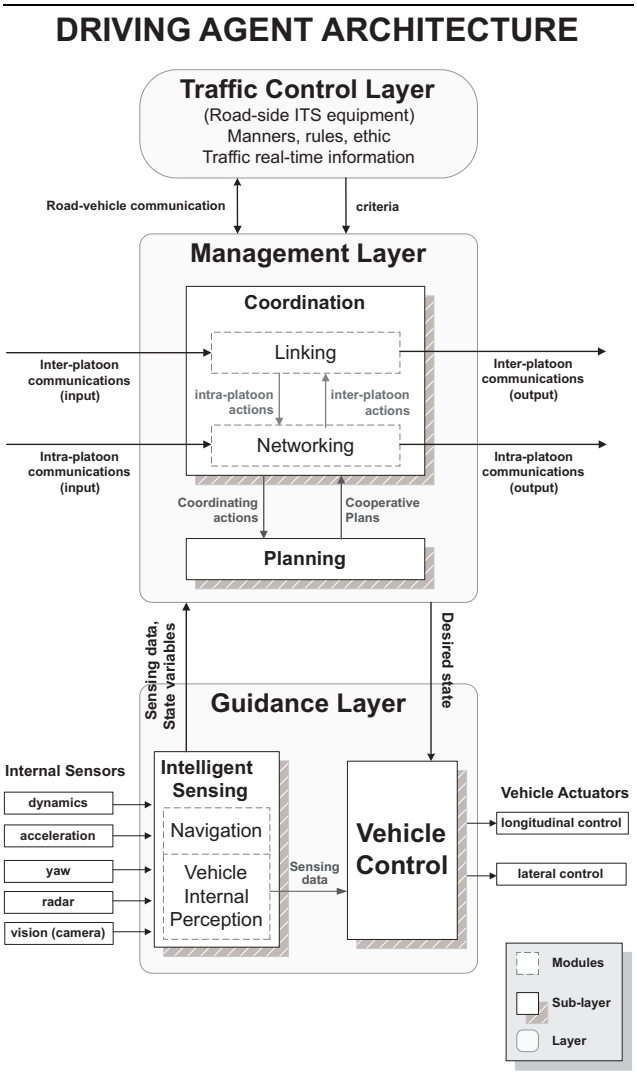

Figure 4. Hierarchical architecture.

using the *networking* module, which is responsible of the intra-platoon coordination and thus, the platoon formation. Finally, the management layer should also maintain a platoon formation plan, a task which is devoted to the *planning* sub-layer.

The *traffic control layer* is a road-side system composed of infrastructure equipments like sign boards, traffic signals and the road-vehicle communications as well as a logical part including: social laws, social rules, weather-manners and other ethics (more specific to Canada), etc.

4. Communication and Coordination methodologies

Communication has shown its value in Collaborative Driving Systems (CDS), by providing faster response time, more efficiency and safety [31], but we must define the most efficient way of using it, in order to take full advantage of this technology. The different possible communication methodologies for the platoon of vehicles are implemented in the *coordination* sub-layer of Figure 4. For the coordination of the platoon and its two main manoeuvres: split, merge, we describe four models and outline their differences in section 5. The models we decided to compare were taken in part from projects as PATH [30], which have mostly used platoon architectures centralized on the leader, although some decentralized models were presented [6]. But we bring this decentralization even forward, to finally come with a novel approach to inter-vehicle coordination in CDS: teamwork for driving agents. We must also specify that in each of these approaches, the guidance and control systems are decentralized for every vehicles involved [11].

Four possible coordination models are presented on Figure 5, starting by the simplest centralized concept, followed by architectures acquiring more autonomy, and thus, decentralization through teamwork theories. Figure 5 highlights the inter-vehicle communications involved in each model, for both the split and merge tasks.

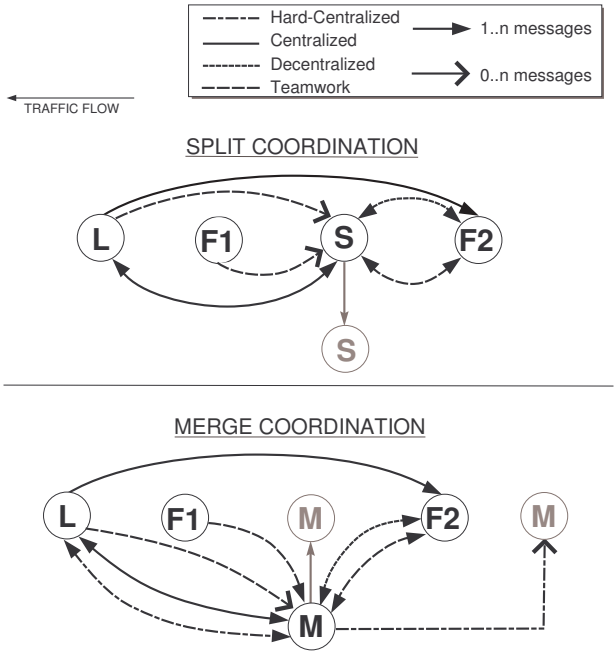

Figure 5. Four coordination models of the merge and split task.

4.1. Centralized Platoons

A centralized platoon means that the task of communication executed to coordinate the vehicle formation is centered on one vehicle: the leader. In this case the leader is the head vehicle of the platoon, and as mentioned earlier, this vehicle is driven by a human (simulated) in our first phase of development. To maintain the platoon formation, the leader is the only entity that can give orders, in which case the followers only apply requested changes. During a split manoeuvre, three vehicles are involved: the leader, the splitter, the vehicle following the splitter (if it exists). During a merge, the same configuration of vehicles is involved: the leader, the merger, the vehicle which will follow the merged vehicle (if it exists). For both of those manoeuvres, the merger or splitter will communicate its need to do a manoeuvre, and then the leader will give requests for inter-vehicle distance, change of lane, meeting point or velocity to involved vehicles. For the merge task, we have defined two sub-models. The first one simplifies the task and involves only two vehicles, by requesting the merging vehicle to always merge at the end of the platoon. In a second model, the leader will specify the optimal in-platoon merging position, considering the merging vehicle's position (parallel to the platoon). Thus, this model will involve three vehicles, if the merging vehicle's position is in front or farther than the platoon's tail vehicle.

4.2. Decentralized Platoons

In the concept of a decentralized platoon, the leader is still the platoon representative, but this is only for inter-platoon coordination. Thus, every platoon member has a knowledge of the platoon formation and is

able to react autonomously, communicating directly with each others. An agent's common knowledge is initialized when it enters the platoon and is updated using the broadcasted information about new vehicles' merge or split (done at the end of such a tasks).

This model represents the simplest decentralization approach and does not rely on any existing framework or complex distributed plans, but tries to lower the communications as much as possible with its simplicity. In this model, the leader is only in charge of maintaining the task safety by notifying others of any emergencies, similarly to the centralized approach. For the split manoeuvre only two vehicles are involved: the splitter and the vehicle following the splitter (if it exists). For the merge, once the merging vehicle has chosen a platoon, only two vehicles are involved as well: the merger, the vehicle which will follow the merged vehicle (if it exists). For those manoeuvres, we eliminate the intermediate that was the leader because every platoon members have the knowledge of its platoon configuration. Meaning that the vehicle following a splitting vehicle knows that it is his task to create a safe inter-vehicle distance. The same applies to the vehicle which is the closest to the merging vehicle, which will react by informing the merger about the right position to merge in and by creating a safe merging gap.

4.3. Teamwork for Platoons

The decentralized model, previously presented, leads us to a more organized decentralized concept, which is the one of teamwork, gaining in popularity in the field of Multiagent. This concept brings a more organized structure to the decentralization of our platoon and provides utilities to maintain safety in the manoeuvres accomplishment. Using Team Oriented Programming (TOP) [19] models, like STEAM [27], the platoon members are assigned roles within a team hierarchy, and team operators relating to those roles are defined, in the same way as other agent's plan architectures [23]. The STEAM architecture also provides domain-independent directives to support responsibilities and commitments for teamwork. Thus, the teamwork strategy results in most vehicles of a platoon to be involved in tasks and communicate if necessary, as shown on the dotted lines of Figure 5, representing "possible" communication.

For the Auto21 project, we have defined three major teams: the platoon formation, the split task team, the merge task team. The first team, is a persistent team, using persistent roles, for long-term assignments as it is the case for the platoon formation. The two latter are task-teams using task-specific roles, for shorter-term assignments, as those teams will not exist after the task completion. Figure 6 illustrates the formation used for a split task team, where the leaf nodes represent roles and in this case, the only internal node, for the task observers, represents a sub-team. Moreover, Figure 7 depicts the operators used by these formations in a tree, similar to Soar's plan hierarchy [23]. In the tree's hierarchy, team operators are surrounded by `[]`, while the other operators are standard individual operators. Those operators only define domain level plans, as the coordination plans are handled by the STEAM infrastructure. The mapping between the organization hierarchy and the task hierarchy is done through the previous role definition [27]. Thus the role assignment of a driving agent will constrain his actions to the operator hierarchy.

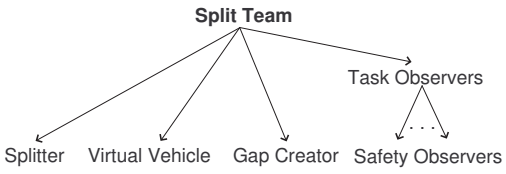

Figure 6. Split task team's role organization.

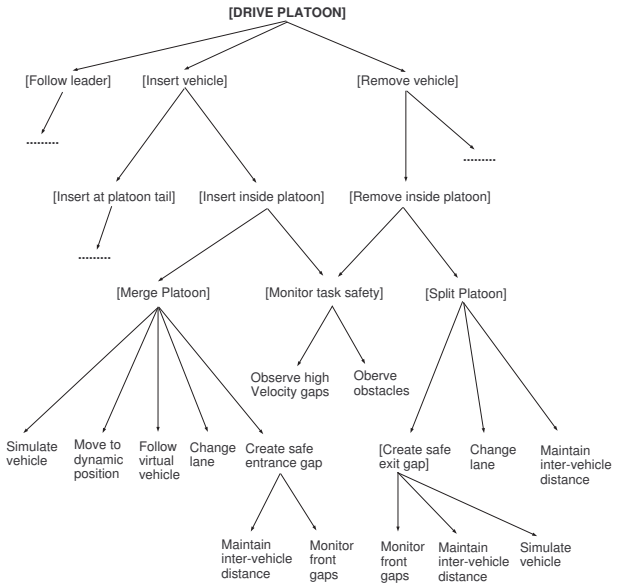

Figure 7. Team and individual operators hierarchy for the platoon formation.

Before going further in the team and task-teams definition, let us explain how the STEAM infrastructure will support the role and operators hierarchy. STEAM is based on the joint intentions theory [15] and thus, the precedent team operators can be seen as intentions on which team members must commit, to begin their execution. In our case, the joint intention of inserting a vehicle in the platoon comes when the members of a merge task-team A mutually believe that a vehicle i wants to merge the platoon. Thus in the joint intention model, this situation could be denoted as $JPG(A, \texttt{[Insert Vehicle]})$, meaning that the task team A has the joint persistent goal of achieving the team action of inserting a vehicle in the platoon. In the precedent context, the joint intention model also defines the precondition of the team plan `[Insert Vehicle]`, which is: the vehicle i is not currently in the platoon, and its postcondition, which is: the vehicle i is in the platoon. Furthermore, since the knowledge of this precondition implies the formation of the task team A, the mutual belief of the postcondition will also imply the end of team A. The joint intention model also specifies a protocol to establish mutual belief, known as the request-confirm protocol [24]. But considering that in our case, the mutual belief will come from a broadcasted (to the intended platoon) communication from the task initiator, we do not need such a protocol.

To assure the team operators' execution, the Team Oriented Programming (TOP) infrastructure will force team members to hold this operator and assure its coherence using STEAM's Coherence Preserving (CP) actions [27]. A CP action, seen as a communicative act to inform others, will be used if the current operator is believed to be unachievable, achieved, or irrelevant. Another type of actions supported by TOP are the monitor and repair actions which are used to preserve the role constraints within the team. Those constraints can be specified as a combination of AND ($\bigwedge$), OR ($\bigvee$), or Role dependency ($\Longrightarrow$), which defines logical relationships during the application of a team's sub-operators, relating to the agent's role. Thus, the achievement of the team operators will be monitored using the logical relationships of its sub-operators.

Within the CDS domain, we have defined three main teams, where the persistent team is the platoon formation. For this formation we only require two persistent (long-term assignments) roles:

- *A leader* which is filled by the head vehicle and who mainly communicates with others using Coherence Preserving (CP) actions. Since the goal here, is to maintain a stable platoon formation, an unsafe deceleration can be seen as a percept that could endanger the goal achievement, therefore influencing the leader to inform others of this fact using CP actions. The probability of such a communicative act will be discussed later.
- *Followers* is a role filled by all the platoon members that are not at the head. In the current scenarios, each vehicle is only equipped with a front laser sensor, so the followers do not need to communicate their percepts to the team. But other sensors, as a rear sensor for the tail vehicle, would bring the followers to possible communications. Thus, when filling this role, a vehicle only maintains the safe inter-vehicle distance.

The merge task-team being similar to the split task-team, we will only depict the merge. This team is centered around the `[Insert vehicle]` team operator, shown in Figure 7, which is executed either as an insertion at the end or an insertion within the platoon. The insertion at the tale being simpler, we did not detailed its branch in the operators' hierarchy tree. As shown in Figure 6, there are four different roles and one sub-team involved in the split (similar to merge), which are detailed as follows:

- *A Merger* is the role filled by the agent initiating the merge team by broadcasting its will to merge a platoon (vehicle *L2* in Figure 3). The operators restricting the merger's actions are the ones within the `[Merge Platoon]` team operator. The `Move to dynamic position` operator will be used by the merger when the task-team has the belief about the entry position for the merging vehicle. This role also uses the `Follow virtual vehicle` operator, which is a virtual representation of *L2*'s future preceding vehicle (*F1*). This virtual vehicle is followed by *L2* before it actually senses the real vehicle with its laser. Finally, the `Change Lane` operator is used here, to switch to the platoon's lane and complete the merge. When the merger is stable in the platoon, a CP action is broadcasted in order to manifest the achievement of the team's goal. Considering those operators, the *merger* role has an AND relation (as defined in STEAM) with the rest of the team, since its performance is crucial to this manoeuvre.
- *Gap Creator* is a role taken by the agent driving the vehicle behind the merging position, in the platoon (vehicle *F2* in Figure 3). Within this role an agent defines the entry position for the merger, since the vehicle it drives will be behind the merger after the lane change. This role requires its filler to execute the `Maintain inter-vehicle distance` operator, which maintains a distance large enough to safely fit a vehicle. Then it has to execute the `Monitor`

`front gaps` operator when the merger is changing lane. A high gap between the last front vehicle percept reading will indicate the arrival of the merging vehicle in the platoon. This will be followed by a new inter-vehicle distance goal. This role's operators are included in the `[Merge Platoon]` team operator, since the aforementioned individual operators are directly linked with the merger's operators execution. For the role relations concerns, this role is also in an AND relation.

- *Virtual Vehicle* is a role that was introduced to assure a stable task execution. This role helps the task executor, when it is in a different lane, to follow the vehicle that was or will be in front of it. In the split and merge tasks, this role is taken by vehicle number *F1* from Figure 3. Within the `[Split Platoon]` team operator, this role applies the `Simulate Vehicle` operator that results in the communication of information about its velocity, if it is modified after the splitting vehicle has changed lane and before the split task is over. Within the `[Merge Platoon]` team operator, the same operator will be applied after vehicle *F1* has transmitted an initial representation of itself to the merging vehicle. This role thus ensures a safe entrance of the merger and eliminates the need to create a virtual representation of the merger to help its future preceding vehicle (*F2*). Indeed, since vehicle *F2* is filling the *Gap Creator* role, which forces him to respect a safe gap with *F1*, and the merging vehicle is also keeping distance with the virtual representation of *F1*, both have the same reference. Thus eliminating the need to create a virtual vehicle for the *Gap Creator* (*F2*). As its predecessor, this role is in an AND relation for the `[Insert platoon]` operator.

- *Safety Observers* is a role taken by one or more agents. The constraint on the role fillers, is that they must be in a position ahead of the task executor's position, so they can monitor dangers in advance. Using the communication selectivity presented next, agents in this role will communicate their belief about dangers or unsafe deceleration to others, taking in account the dangers of sudden movements during a task execution. Agents filling in this role conjointly execute the `[Monitor task safety]` safety team operator, therefore executing observation plans individually. This role can be filled in by multiple agents, which have an OR relation. The combination of the *Safety Observers*' operators with th rest of its team is done using a ($\Longrightarrow$) role dependency. This means, that the execution of the three precedent roles is crucial to achieve the goal and maintain this role, but the execution of the *Safety Observer* role is not critical.

The Selective Communication (SC) actions taken from STEAM, is close to Tambe's latest infrastructure: COM-MTDP (**Com**municative **M**utiagent **T**eam **D**ecision **P**roblem) [20]. The latter infrastructure considers communication selection's optimality, as opposed to STEAM, which uses decision-theoretic communication selectivity. These SC actions are used to synchronize mutual beliefs within the execution of team operators. A SC action thus verifies if a communication within a team must be done, according to the domain's communication costs and benefits. Not only that, but the selective communication also verifies the likelihood that the information it wants to communicate, is already common knowledge. For the precedent merge task-team, the agent that fills in the *Virtual Vehicle* role will communicate changes to its virtual representation (a new velocity for example) if it believes that the merging vehicle does not pick him up on its sensor and that this velocity change is important enough. To illustrate this situation, Figure 8 shows a decision tree representing the Selective Communication decision the agent must take, as part of the STEAM framework [27]. The first 2 branches represent the choice of communicating or not, which are then divided by the probability ρ that the information (belief) it wants to communicate is not known by its teammates. Furthermore, a third pair of branches is added to specify the probability σ that this information opposes a threat to the execution of the current team operator. This last probability represents the significance of the new information compared to the current mutual belief, for the current team operator. As opposed to the general use of SC actions proposed in STEAM, we will not use them for task-team formation and dissolution. This means that SC actions will not be used for the Coherence Preserving (CP) actions, which notify of the `[Insert vehicle]` and `[Remove vehicle]` operators' achievement or creation. Thus, SC will be used to apply a selection over the communication sent to maintain mutual beliefs during the execution of a team operator. By relating to the previous example using the *Virtual Vehicle*, this agent will use the tree defined in Figure 8 to make a decision on wether it will communicate an update on its position, thus synchronizing the team's belief on its virtual representation. To make this decision it must verify if the expected utility of making this communication $EU(C)$ is higher than the expected utility of not communicating $EU(NC)$. $EU(C)$ is defined as:

$$EU(C) = S - (Cc + (1 - \rho) * Cn)$$

Which is a reward S for synchronization of the team's belief during the execution of a team operator, minus the Cost of communication Cc and the Cost of nuisance Cn. Cn is used in probability $1 - \rho$, which is the probability that the information to communicate is already known by the team. $EU(NC)$ is defined in the same way:

$$EU(NC) = S - (\rho * \sigma * C_{mt})$$

Where the Cost of nuisance is replaced by C_{mt}: the cost for miscoordination. These two definitions give us the equation to make a decision on the Selective Communication, thus communicating when the $EU(C) > EU(NC)$, i.e., iff:

$$\rho * \sigma * C_{mt} > (Cc + (1 - \rho) * Cn)$$

In the merge example, the cost for communication is higher than normal, since more communications are involved during a manoeuvre and we do not want to saturate the network. Then, if the vehicle *F1* has to modify its velocity during the merge, the probability ρ that this new information on *F1*'s velocity is commonly known, will mainly depend on the probability P(*L2*,*F1*) that the merging vehicle *L2* has *F1* in its sensor's range (if it is in the platoon). Furthermore, probability σ that this information opposes a threat to the merge manoeuvre depends on the difference between *F1*'s knowledge about its velocity and the team's belief. If the team is highly out of synchronization, the agent will communicate at a higher probability. Finally, the C_{mt} and Cn costs will be set to an average-low value for this task.

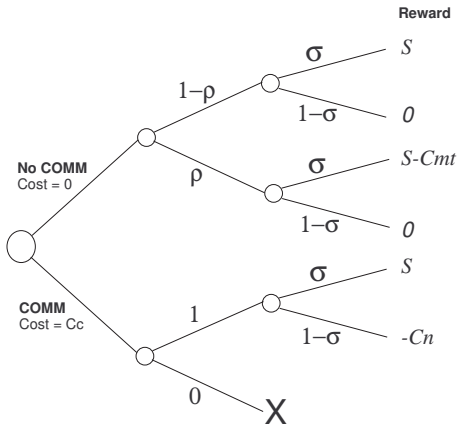

Figure 8. Decision tree with probability and rewards for communicative acts.

This kind of decision-theoretic selector being part of the TOP framework, we will be using it within all of the teams and roles presented earlier. Team knowledge relating to sub-operators' pre/post-conditions will be awarded a great value for σ and C_{mt} to insure the communication of this type of information, even though agents may have doubts on the team's belief about it. Moreover, these probabilities should be adapted through testing, like it has been done for domains like the RoboCup soccer challenge [18], where data traces of team behaviors were used to learn probabilities distribution. This approach proposes an offline learning approach on patterns of communication within the team, that can be applied to specify the probabilities of the SC operators.

5. Evaluation and Results

To develop the previous theory on coordination strategies, we have used an agent development toolkit called JACK Intelligent Agents^{TM}[1] which supports the Belief Desire Intention (BDI) agent model [21], as well as teamwork related strategies. In the TOP vision relating to the STEAM model, a non-negligible advantage is the reusability and flexibility of the operators [27], since it contains many infrastructure rules that are not directly related to the domain level. Thus, using JACK's vision of agent's capabilities, related to plans and beliefs, we managed to develop collaborative driving teams that follow TOP models. It must be mentioned that Agent Oriented Software (AOS), who develops JACK, also developed an extension called JACK Teams^{TM}that supports the vision of shared plans and beliefs, and team formation using roles. Although this extension to JACK looks promising, it is not complete at the moment and lacks a lot in documentation to realize dynamic task-team formation as described in this paper. Therefore, we have developed our own TOP extension to Jack, although it will possible in a near future to merge to AOS' new extension.

The coordination models presented in section 4 have been implemented according to the architecture presented in Figure 4. We show as an example in Figure 9, the results we got in the average coordination of a vehicle exit (split manoeuvre), using a centralized coordination (in a thin red line) as opposed to the teamwork model of coordination (in a bold blue line). This graphic shows results using the splitter's (vehicle *F2* in Figure 3) data and the splitter's preceding vehicle's (vehicle *F3*) data. The precedent vehicle senses the splitting vehicle and has to adjust from its departure by keeping a *safe* gap until the splitter is safely out. The solid lines present the difference between, the front dis-

tance between this vehicle and its front vehicle (splitter), and the *safe* front distance. This *safe* distance is defined by a gap in time between vehicles that agents should respect to insure security. In addition, the dotted lines only show the inter-vehicle distance from *F3*'s sensor, without applying a difference. Around time 14, vehicle *F3* has to create a larger and safer distance with the splitting vehicle, so the solid lines drop, but are readjusted within almost 10 seconds. The second outlined step arrives at time 30, and 27 for the teamwork, when the splitter has went out of range of vehicle *F3*'s laser. At this moment, the sensed distance raises on the dotted line, but there is no gap considering the distance defined as *safe* (solid lines). Before the splitter has stabilized itself on the next lane (time 37 or 34), the gap creating vehicle of the centralized model does not manage to keep the *safe* distance and has a difference of 2 meters with the *safe* distance by the end. On the other hand, the splitting vehicle modelled with the teamwork coordination is using communications from vehicle *F1* through teamwork rules, to maintain his virtual representation and follow it after it has changed lane, thus helping its following vehicle to maintain the right *safe* distance. This approach gives much better results, since the difference with the *safe* distance does not go higher than 0.5 m. When the splitter is stabilized, the distance qualified as *safe* drops to the normal intra-platoon distance. At that moment, the vehicle is at 17 meters (length of the gap created by the vehicle that left) from the *safe* distance, reached by the end. This graphics also shows that using a teamwork model, information is exchanged faster since messages do not have to go through the leader, which results in an overall faster response time of three seconds by the end of the task.

The implementation of four different coordination strategies leads us to conclusions concerning their respective advantage and disadvantage. First, as mentioned in section 4.1, a fully centralized coordination at its simplest version was developed. This approach is centralized on the leader and only allows the merging vehicle to enter at the platoon's tail. Second, we developed another model which was centralized on the leader, but allowing entrance anywhere inside the platoon. In the third approach, the coordination is decentralized, so the vehicle executing the manoeuvre only has to coordinate itself with the vehicle directly concerned by this task. Thus, a splitting vehicle will only communicate with its rear vehicle to create a gap and leave the platoon, and a merging vehicle will only exchange messages with the closest platoon member vehicle, so it can create a gap for him to merge. The fourth approach uses the previously detailed teamwork model.

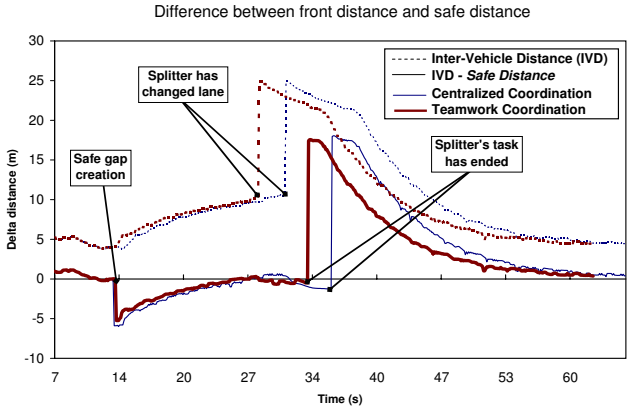

Figure 9. Inter-vehicle distance with the splitting vehicle.

Strategies on decentralized approaches are still under development and extensive simulations including multiple platoons to increase traffic density will have to be done to improve our results and the choice being done in the end. But using the preliminary results, presented in Table 1, we show each of the four models used for both a split and a merge, divided in four rows. We compared on the first pair of column, the average total amount of messages exchanged by vehicles during the manoeuvre. On the second pair, we compare the amount of plans taken to develop the *Coordination layer* of our architecture, shown in Figure 4. Those plans were defined using JACK^{TM}[1], and represent contextual rules of applications for communication and actuation. The amount of messages calculated for the merge task consider that the merging vehicle had already chosen his platoon and is ready to merge. Those four models advantage and disadvantage are summarized as follows:

1. The first centralized model (*Hard-Centralized*) is very benefic on the amount of messages it exchanges. But the major disadvantage is the traffic density it creates, as it must reach the platoon's tail by either accelerating or decelerating (considering his position), thus creating traffic waves and diminishing the highway's capacity.

2. The second centralized model suffers from the amount of messages it encounters, as the leader redirects all the messages within the platoon. Moreover, in average, more than three quarters of the messages were sent or received by the leader, creating a bottleneck for this vehicle. The centralized model does not require the followers to keep

a platoon knowledge, which helps lowering communications compared to decentralized models. As for the previous model, the centralized coordination uses static coordination protocols supported by the leader, which has the disadvantage of not allowing much flexibility on the coordination of unexpected situations.

3. The standard decentralized approach uses less messages but is a lot less safer than the other approaches. This fact will be proven using further simulations, but since it always uses only two actors, and no virtual vehicle (as mentioned for the teamwork model in 4.3), this approach would have to compensate by using more sensors to attain the level of safety of the teamwork model. This model also needs to communicate to initialize and maintain common knowledge within the platoon, but since the "updates" on knowledge are done through a broadcast, it does not require much more messages.

4. In the teamwork model, we managed to use an amount of messages that is in the average of the three other approaches. It must be mentioned that this number varies more than in the other models, from different contextual simulations, because of the selective communications. Furthermore, only the teamwork model implemented the virtual vehicle, which explains a higher number of messages. Then, using a TOP model implemented with the STEAM vision in JACK, even though there are more actors for a safer approach, we do not need much more plans than the standard decentralized approach, which was not developed using generic plans. Moreover, TOP uses less plans than the decentralized model, developed in a more functional vision, in which more plans are required to handle uncertainty, compared to STEAM's vision. Compared to the decentralized model, the TOP framework is in charge of the platoon's belief (common knowledge) and manages to handle better the communications required for this matter. At last, as it was mentioned for the decentralized model, the advantage of this coordination model would be attenuated if the vehicles' communications would not be done through broadcast, since we need to maintain common beliefs within the platoon.

6. Conclusion and Future Work

Collaborative driving is emerging in the ITS domain and its need for communication is obvious. As this paper presented, building an autonomous driving system on a strong architecture giving a wide latitude to the coordination strategies enables us to use different inter-vehicle communication models. From these models, we have presented and tested four different strategies, for which advantage and disadvantage were presented. The coordination model based on teamwork presented great avenues considering the flexibility and safety insurance at low communication cost. In addition, the decision theoretic communication selection allied with STEAM's Coherence Preserving actions enables us to define a more generic view of the plans, and provides code reusability and flexibility.

	Nb Messages		Jack Plans	
Coordination	Merge	Split	Merge	Split
Hard-Centralized	7	7.5	12	12
Centralized	11.5	8	20	13
Decentralized	8	5.5	12	9
Teamwork	8.75	6.75	14	10

Table 1. Total of messages and plans used by coordination model

The Collaborative Driving System presented in this paper could be used in a fully autonomous system, using vehicles equipped with longitudinal and lateral guidance system. But within the presented scenarios, we did not specify if the lateral control was automated or a simulation of a human driver, thus we are are still opened to both avenues. This collaboration system could then be used in a navigation system inside a car as an Adaptive Cruise Control, which would more easily acquire the public's favor. Furthermore, these collaborative driving strategies will be used within a simulation of the Canadian climate environment, thus demonstrating the application of CDS to snowy roads, conjointly with the required ITS infrastructure.

We will continue improvements on the longitudinal guidance system that could enable us to lower the communication probabilities in the selective communication decisions of the Team Oriented Programming (TOP) infrastructure. The different coordination strategies will be further extended and many more scenarios involving uncertainty will be taken into account using the simulator. We will look at RMTDP role reallocation strategies and machine learning applied to decision making on communications, to improve the results involved in those new simulation scenarios.

7. Acknowledgments

This work has been carried out as part of the Automobile of the 21st Century (Auto21) project supported by the Government of Canada through the Networks of Centres of Excellence (NCE).

References

[1] Agent Oriented Software Pty. Ltd. JACK Intelligent Agents^{TM} 4.1. Software Agents Development Framework, 2004.

[2] J. S. Albus. Outline for a theory of intelligence. *IEEE Transactions Systems Man, and Cybernetics*, 21(3):473–509, May 1991.

[3] M. Antoniotti, A. Deshpande, and A. Girault. Microsimulation analysis of a hybrid system model of multiple merge junction highways and semi-automated vehicles. In *Proocedings of the IEEE Conference on Systems, Man and Cybernetics*, FL, U.S.A., October 1997.

[4] Auto'21. Architectures for collaborative driving vehicles: From a review to a proposal. Technical report, Université Laval, Ste-Foy, Québec, 2003.

[5] Auto21. [Online], May 2004. http://www.auto21.ca (consulted the 18th of May 2004).

[6] S. Bana. Coordinating automated vehicles via communication. PATH Research Report UCB-ITS-PRR-2001-20, CS Berkely, 2001.

[7] B. Chaib-draa and P. Levesque. Hierarchical models and communication in multi-agent environments. In *Proceedings of the Sixth European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-94)*, pages 119–134, Odense, Denmark, Aug. 1994.

[8] DAMAS-Auto21st. [Online], May 2004. http://www.damas.ift.ulaval.ca/projets/auto21/, (consulted the 18th of May 2004).

[9] F.Michaud and J. de Lafontaine. Current status, strategy and future work. Fcd-sp-004-udes, Universit de Sherbrooke, June 2002.

[10] S. Ghosh and T. Lee. *Intelligent Transportation Systems: New Principles and Architectures*. CRC Press, 2000.

[11] X. Huppe, J. de Lafontaine, M. Beauregard, and F. Michaud. Guidance and control of a platoon of vehicles adapted to changing environment conditions. In *IEEE International Conference on Systems Man and Cybernetics, 2003.*, volume 4, pages 3091–3096, 2003.

[12] P. Ioannou and M. Stefanovic. Evaluation of the acc vehicles in mixed traffic: Lane change effects and sensitivity analysis. PATH Research Report UCB-ITS-PRR-2003-03, University of Southern California, 2003.

[13] N. R. Jennings. Towards a cooperation knowledge level for collaborative problem solving. In B. Neumann, editor, *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI-92)*, pages 224–228, Vienna, Austria, 1992. John Wiley & Sons.

[14] U. Kiencke and L. Nielsen. *Automotive Control Systems: For Engine, Driveline and Vehicle*. Springer Verlag, March 2000.

[15] H. J. Levesque and P. R. Cohen. On acting together. In *Proceedings of AAAI-90*, 1990.

[16] W. Lin and H. Leung. Comparison of vehicle detectors used in intelligent transportation systems. Technical report, NCE, Auto'21, 2002.

[17] J. Lygeros, D. N. Godbole, and S. S. Sastry. Verified hybrid controllers for automated vehicles. In *IEEE Transactions on Automatic Control*, volume 43, pages 522–539, 1998.

[18] R. Nair, M. Tambe, S. Marsella, and T. Raines. Automated assistants for analyzing team behaviors. *Journal of Autonomous Agents and Multiagent Systems (JAAMAS)*, 8(1):69–111, Jan. 2004.

[19] D. Pynadath, M. Tambe, N. Chauvat, and L. Cavedon. Toward team-oriented programming. In *Proceedings of the Agents, theories, architectures and languages (ATAL'99) workshop (to appear)*, 1999.

[20] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of AI research*, 16:389–423, 2002.

[21] A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco, 1995.

[22] J. K. Rosenblatt. *DAMN: A Distributed Architecture for Mobile Navigation*. Doctor of philosophy in robotics, Carnegie Mellon University, Pittsburgh, Jan. 1997.

[23] P. S. Rosenbloom, J. E. Laird, A. Newell, and R. McCarl. A preliminary analysis of the soar architecture as a basis for general intelligence. *Artificial Intelligence*, 47(1-3):289 – 325, July 1991.

[24] I. Smith and P. Cohen. Towards semantics for an agent communication language based on speech acts. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1996.

[25] R. R. Stough. *Intelligent Transportation Systems: Cases and Policies*. Edward Elgar Pub. LTD: Cheltenham, UK/Northampton, MA, USA, 2001.

[26] R. Sukthankar, J. Hancock, and C. Thorpe. Tactical-level simulation for intelligent transportation systems. *Journal on Mathematical and Computer Modeling*, 1997.

[27] M. Tambe and W. Zhang. Towards flexible teamwork in persistent teams: extended report. *Journal of Autonomous Agents and Multi-agent Systems, special issue on Best of ICMAS 98*, 3:159–183, 2000.

[28] S. Tsugawa, S. Kato, T. Matsui, and H. Naganawa. An architecture for cooperative driving of automated vehicles. In *Procs. IEEE Intelligent Vehicles Symposium 2000*, pages 422–427, Dearbsorn, MI, USA, Oct. 2000.

[29] S. Tsugawa, S. Kato, K. Tokuda, T. Matsui, and H. Fujii. A copperative driving system with automated vehiclues and inter-vehicle communications in demo 2000. In *Proceedings of the IEEE Intelligent Transportation Systems Conference*, 2001.

[30] P. Varaiya. Smart cars on smart roads: problems of control. *IEEE Transactions on Automatic Control*, 32, 1993.

[31] Q. Xu, K. Hedrick, R. Sengupta, and J. VanderWerf. Effects of vehicle-vehicle / roadside-vehicle communication on adaptive cruise controlled highway systems. In *IEEE VTC*, Fall 2002.