Collaborative Driving System Using Teamwork for Platoon Formations

Simon Hallé and Brahim Chaib-draa

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 achieve this, we propose a hierarchical architecture based on three layers (*Guidance* layer, *Management* layer and *Traffic Control* layer), which can be used to develop coordination models for 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 mainly considers a software agent teamwork model using architectures like STEAM. These different coordination models will be compared using results on preliminary 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 [7]. It is shown that ITS may provide potential capacity improvements as high as 20 percent [14]. 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, which are generally sustained by individual technologies as: electronic sensors, wire and wireless

Received by the editors February 6, 2005.
1991 *Mathematics Subject Classification.* Primary 99Z99; Secondary 00A00.
Key words and phrases. Collaborative Driving, Intelligent Vehicles, Teamwork, Multiagent, Platoon, Intelligent Transport System.
This work was 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).

communications, computer software and hardware, GPS, GIS, etc. To sum up, ITS has the advantage of enhancing safety and reducing congestion, which results in a reduction of transport systems' negative environmental impacts.

Collaborative driving is an important sub-component 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) [11]. Thus, at its simplest implementation, collaborative driving adds 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 [18]. Going forward to a wider level of collaboration, the vehicle platoon model, has used communications to coordinate platoon members with their platoon leader [16]. Compared with CACC, this vehicle organization adds a deliberative system in the lead vehicle, which coordinates the preceding vehicles equipped with CACC, to maintain the platoon formation. As a new approach to centralized platoon models based on the leader, we aim to incorporate the multiagent vision to the platoon architecture and coordinate the vehicles through a model of teamwork for software agents [15]. Such an approach incorporates autonomous agents in each vehicle to use their respective communication system and coordinate each others in a decentralized platoon model.

In this paper, we address in Section 2 the coordination issues for a platoon of vehicles, by first describing the collaborative driving domain and the simulator used to represent this environment. 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 our preliminary results using the comparison of centralized coordination models with decentralized ones, by putting the emphasis on agent teamwork. Finally, Section 6 presents a discussion, followed by the conclusion.

2. Application Domain

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* [17], 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, so other vehicles can simply follow it. Our work comes as part of the Auto21 project [3] studying the automobile of the 21st century within three levels of system functionality. The first level focuses on the vehicle's longitudinal control, while the second focuses on its lateral control, both in a platoon lead by a human driver. Finally, the third level considers every vehicle (even the leader) as fully autonomous. This article focuses on the first level of functionality and its usage of communications within the different platoon driving manoeuvres.

2.1. Microsimulation of Autonomous Vehicles

The environment in which our vehicle coordination system has been tested is a Collaborative Driving System (CDS) [5] simulator developed to provide user interface and graphical results of our work. Similar to California Path's Smart AHS [1], our simulator called HESTIA, aspires to a lower level of vehicle microsimulation, 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. These vehicles 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 that offers a 3D environment in which autonomous vehicles can evolve.

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 simulated sensors were developed using the 3D engine of JAVA 3D^{TM}, and in the vehicle model presented in this paper, each following vehicle is equipped with a vehicle-based laser sensor. This sensor provides information on the front object's (a vehicle) distance and difference of velocity, for distances up to $100m$, 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. We will not go further on a detailed representation of the simulator's components, as it is out of scope for this paper, but the readers can refer to [8] for more information.

2.2. Simulated Driving Scenarios

The primary goal of the CDS is to maintain the platoon formation stable, and therefore, the two simulated scenarios we focus on are the two main disturbance in this formation: a vehicle splitting and a vehicle merging the platoon. These two scenarios are represented in Figure 1 and they 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 splitting vehicle (*F2* in Figure 1) 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 1. When this new formation gains stability, the splitting vehicle *F2* can change lane, while the rest of the platoon followers keep the same distance. When the splitting vehicle has 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. To execute this manoeuvre, the merging vehicle must be part of a platoon formed of one vehicle (itself), like vehicle *L2* in

Figure 1. The merging vehicle first has to communicate to another platoon a query to join it. In the example of Figure 1, in *step 1* (*S1*), the platoon lead by vehicle *L1* accepts *L2*'s query. Moving from *S1* to *S2*, *L1*'s platoon reacts by creating a safe space and communicates the dynamic position of this space to the merging vehicle. The merging vehicle then 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 meet *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 manoeuvre may differ from one coordination model to another, this represents the general pattern of the merge manoeuvre.

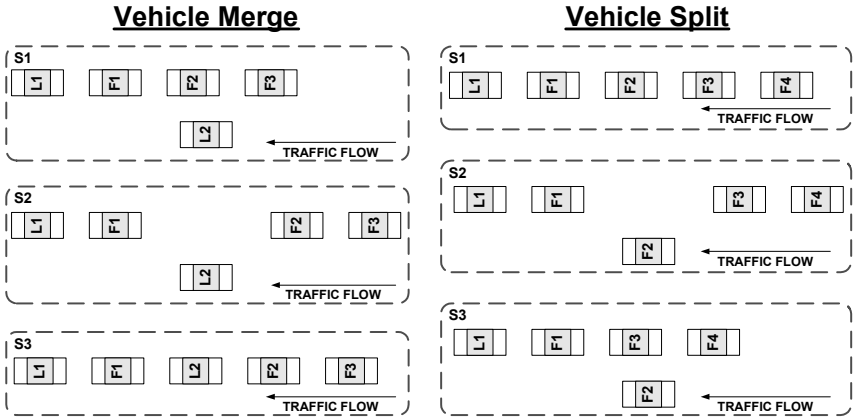

FIGURE 1. The three steps of the removal (split) and insertion (merge) of a vehicle in the platoon.

As it has been shown, the merging and splitting manoeuvres of a vehicle are the most problematic cases of the platoon formation and this is why the different coordination models presented in Section 4 focus on the communications involved during these two tasks. Since we put the emphasis on the communication and coordination of the platoon, the vehicles' lateral automated control is simulated to enable us to perform the lane changes involved in the merge and split manoeuvres. The lateral guidance system of our current system could then be seen as the simulation of the human driver's steering behavior or as the first phase of the lateral guidance system. This subject being out of scope for 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. 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.

Finally, as we related to other collaborative driving models, our hierarchical architecture was also inspired by concepts coming mainly from the PATH project [12]. As indicated in Figure 2, the resulting architecture has three major layers: *Guidance* layer, *Management* layer and *Traffic Control* layer.

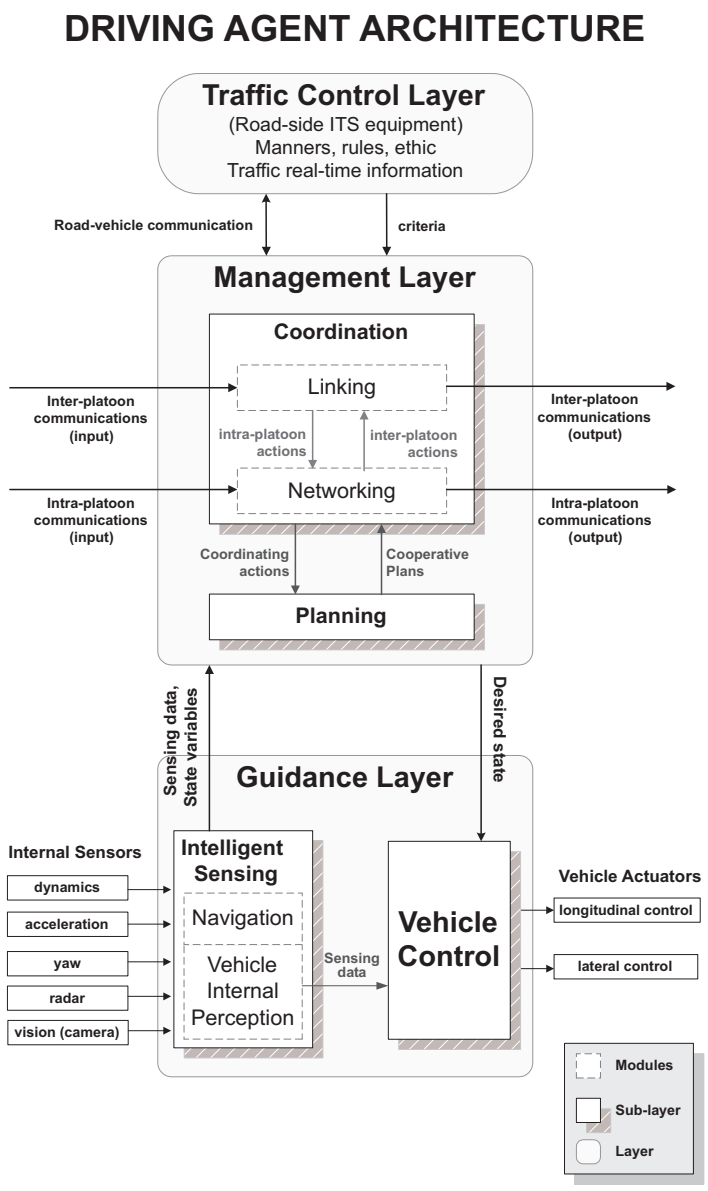

FIGURE 2. Hierarchical driving architecture.

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 *Intelligent Sensing* sub-layer, the inputs come from sensors for speed, acceleration, raw rate, machine vision, etc. The *Guidance* layer outputs sensing data and vehicles state variables to the vehicle *Management* layer, which in turn sends back "desired state" queries in the form of steering and vehicle velocity queries to the *Guidance* layer. These queries are finally applied by the *Vehicle Control* sub-layer, which includes lateral controllers and the longitudinal controllers developed by our partners at Sherbrooke University [10].

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; and (c) the *Traffic Control* layer through the road-vehicle communication. To determine the movement of each vehicle under the cooperative constraints, the *Management* layer needs to reason on the place of the vehicle in its platoon when the vehicle stays in the same lane (intra-platoon coordination), and its place in a new platoon when the vehicle should change lane (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 architecture's *Linking* module reasons about the place of its vehicle on the highway and it coordinates lane change actions with neighboring vehicles (inter-platoons coordination). Following the synchronization of neighbors' lane change requests, the vehicles executing their lane change have to coordinate the actions involved in the lane change. For example, if a vehicle leaves a platoon to enter a new one during its lane change, it has to coordinate both a split and a merge manoeuvre. This coordination task is handled by the *Networking* module, which is responsible of the intra-platoon coordination and thus, the platoon formation's stability. The *Networking* module coordinates driving actions with other driving agents involved in the manoeuvre (split or merge) to finally plan a series of local actions using the *Planning* sub-layer. This sub-layer schedules driving actions (in time or according to events) that are locally executed by sending "desired state" queries to the *Guidance* 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 by enhancing safety [18], 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 2. 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 [17], which have mostly used platoon architectures coordinated by a master entity (the leader), although some decentralized coordination models were also proposed [4]. However, in our application, we bring this decentralization even forward, to finally come with a novel approach to inter-vehicle coordination in CDS: a teamwork coordination for driving agents. In this section, the centralized, decentralized and teamwork models are described by focusing on the teamwork model, which is more complex and represents the most promising research avenue for our application. Note that in each of our coordination models, the guidance and control systems are decentralized for every vehicle involved [10].

Figure 3 presents the four coordination models we compared inside a common CDS framework. This figure illustrates the differences in the communicative behavior of each model by showing which vehicles are involved in the split and merge manoeuvres, and wether each vehicle "surely" communicates or not. Accordingly, Figure 3 outlines the fact that every communication in the centralized model involves the leader, while the communications in the decentralized model only involve two vehicles. Moreover, the communications in the teamwork model involve most of the platoon members, but they do not necessarily communicate during the manoeuvre, as it is shown in Section 4.3.

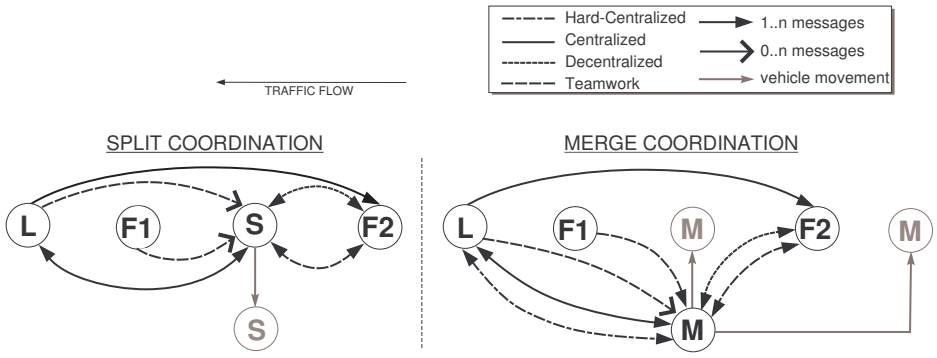

FIGURE 3. Four coordination models of the merge and split task.

4.1. Centralized Platoon Coordination

In a centralized platoon coordination model, the communications are centered on one "master vehicle" giving orders to the rest of the platoon: 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 splitting vehicle, the vehicle following the splitting vehicle (vehicle *F3* in Figure 3). During a merge, the same configuration of vehicles is involved: the leader, the merger, the vehicle which will follow the merged vehicle after its lane change (vehicle *F2* in Figure 3). For both of these manoeuvres, the merger or splitter first communicates its need to do a manoeuvre, and then, the leader communicates requests for inter-vehicle distance, change of lane, meeting point or velocity to involved vehicles.

For the merge manoeuvre, we have defined two sub-models: hard-centralized and centralized. The hard-centralized model simplifies the task and only requires two vehicles to communicate, by requesting the merging vehicle to always merge at the end of the platoon. In the centralized model, the leader specifies the optimal in-platoon merging position, considering the merging vehicle's position (parallel to the platoon). Thus, the centralized model requires three vehicles to execute a merge (leader, merger and gap creator), while the hard-centralized model requires only two (leader and merger).

4.2. Decentralized Platoon Coordination

In a decentralized platoon coordination model, 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 manoeuvres).

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. In this model, the leader is only in charge of maintaining the platoon 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 (vehicle *F2* in Figure 3). 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 after its lane change (vehicle *F2* in Figure 3). For these two manoeuvres, we eliminate the intermediate, i.e., the leader in the centralized model, because every platoon member has the knowledge of its platoon configuration. To replace the leader and coordinate platoon members without relying on a single vehicle, we used a set of social laws, which specify the state in which an agent can take part in a manoeuvre. This way, the actions of creating a safe gap for the split and merge tasks can be handled by a platoon member without being assigned by the leader. For instance, the intention of creating a safe gap emerges (through social laws) from the vehicle at the right distance from the merging vehicle, while the other platoon members determines that it is not their task.

4.3. Teamwork for Platoons

The previous decentralized model can be improved using a better structured organization, as the teamwork for agents, a concept gaining in popularity these recent years, in the filed of multiagent systems. In this context, a teamwork architecture like STEAM [15] can be used to assign roles to platoon members within a predefined team hierarchy. Using this type of architecture, most of the communications required to coordinate team members are handled by the framework. Thus, the teamwork concept results in most vehicles of a platoon to be involved in tasks and communicate when it is necessary, as shown by the dotted lines of Figure 3, representing "possible" communication. For the Auto21 project, we adapted STEAM's communication framework (based on STEAM operators) considering our specific needs. We then defined the required CDS team structures and developed a set of driving plans as domain-level operators, which can be used inside STEAM's framework. This section presents the previous aspects starting with the Auto21 team formations and the Auto21 domain-level operators, followed by the description of STEAM's operators, which are used to ensure coherence inside our team formation.

4.3.1. Auto21 Team Formations & Operators For the Auto21 project, we defined three major teams: (i) the "platoon formation" team; (ii) the "split task" team; and (iii) the "merge task" team. The "platoon formation" team is a persistent team using persistent roles for long-term assignments. The two latter types of team are task teams using task-specific roles, for shorter-term assignments, since these teams stop existing after the task completion. Each of these teams is formed of different agents that must fill all the roles that are specified in the team's definition. For instance, the "split task" team is defined by Figure 4's tree, in which the leaf nodes represent roles and the internal node (only one in this case) represents a sub-team (the task observers). When an agent fills a role inside a team, this role defines the actions this agent can execute, while the notion of team defines the agent's goal or intention.

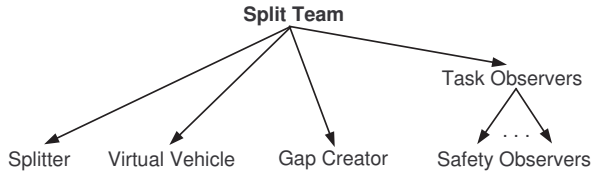

Figure 4. Split task team's role organization.

Following the assignation of a specific role to an agent, the STEAM framework constrains this agent to execute domain-level operators that are specific to the assigned role. Domain-level operators refer to the agent's actions (programmed as plans in our application) and they can be defined in a hierarchial tree. Such a tree is presented in Figure 5, which depicts the operators used by the merge team

formation of our CDS. In the tree's hierarchy, team operators are surrounded by `[]`, while the other operators are standard individual operators. Agents evolving in the STEAM framework are able to execute two possible types of operator: domain-level; and architecture-level (STEAM operators). The operators presented in Figure 5 are from the domain-level, since they represent actions of the CDS domain, while the architecture-level operators, described in Section 4.3.2, are independent of the application's execution environment.

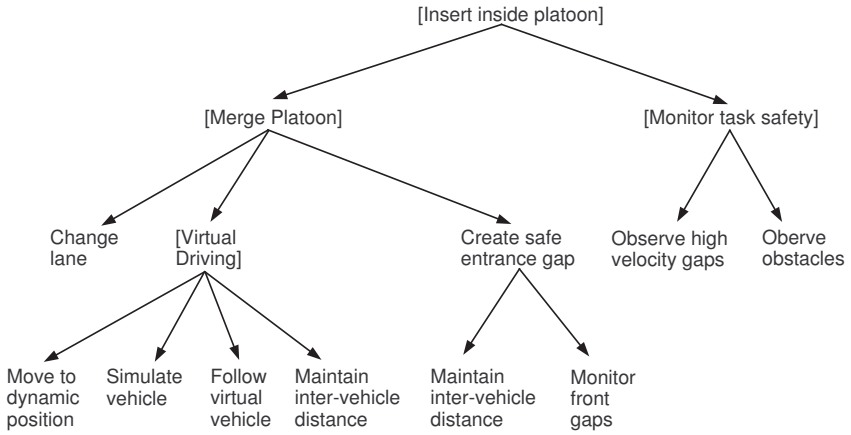

FIGURE 5. Team and individual operators hierarchy for the merge manoeuvre.

As mentioned above, we developed three types of team for the Auto21 project. The "platoon formation" team is the simplest team, where each driving agent holds the intention of maintaining a stable and safe platoon formation. At the moment, we consider that each member of the "platoon formation" has the same task, which consists of following the front vehicle in a safe manner. Hence, this formation only requires two persistent (long-term assignments) roles:

- *Leader:* a role filled by the head vehicle, which mainly communicates with others using Selective Communication (SC) operators (defined in Section 4.3.2). Since its goal is to maintain a stable platoon, the leader can perceive an unsafe deceleration as a situation that could endanger the goal achievement, therefore influencing the leader to inform others of this fact using SC operators. The probability of this communicative act is discussed later.
- *Follower:* a role filled by all the platoon members that are not at the head. At the moment, each follower's goal consists of maintaining the safe inter-vehicle distance with the preceding vehicle, in order to maintain the platoon's stability. An agent in this role does not need to communicate any information since the automated driving system of each vehicle is capable of maintaining the platoon stable in the context of a "platoon formation" team. Thus, the task of maintaining a safe distance is realized by using the vehicle's front sensor and possible information from the leader.

The "merge task" team being similar to the "split task" team, we only depict the "merge task" team in this paper. This team is centered around the [Insert vehicle] team operator, shown in Figure 5. As shown in Figure 4, the split team (similar to the merge team) is formed of four different roles and a sub-team: the *Task-Observers* sub-team. Each role is described below by referring to the operators they use in Figure 5's tree and the place they occupy in the platoon, in Figure 1's illustration of the merge manoeuvre.

- *Merger:* a role filled by the agent which initiates the "merge task" team by broadcasting its intention to merge a platoon (vehicle *L2* in Figure 1). The merger executes the [Merge Platoon] team operator (refer to Figure 5), which restricts its local operators to the ones below [Merge Platoon], in the same tree. The Move to dynamic position operator is used by the merger when the "merge task" team has acquired a mutual belief about the merger's entry position in the platoon. The *Merger* 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.
- *Gap Creator:* a role filled by the agent driving the vehicle behind the merging position, in the platoon (vehicle *F2* in Figure 1). Within this role, an agent defines the entry position for the merger, since its vehicle will be behind the merger after the lane change. The *Gap Creator* role requires its filler to execute the Maintain inter-vehicle distance operator (refer to Figure 5), which maintains a distance large enough to safely fit a vehicle. Following the execution of this operator, the *Gap Creator* has to execute the Monitor front gaps operator when the merger is changing lane. This operator monitors high gap values between the last front vehicle percept reading, to conclude on the arrival of the merging vehicle in the platoon. At the end of the merge manoeuvre, the *Gap Creator* executes the Maintain inter-vehicle distance with a smaller distance value, in order to close the gap in the platoon.
- Virtual Vehicle: a role that was introduced to ensure a stable execution of the manoeuvre. This role helps the manoeuvre executor (splitter or merger), 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 manoeuvres, this role is taken by vehicle number *F1* from Figure 1. Within the [Split Platoon] team operator (not illustrated here), the *Virtual Vehicle* role applies the Simulate Vehicle operator if its own velocity is modified after the splitting vehicle has changed lane and before the split manoeuvre is over. This operator results in the communication of information about the *Virtual Vehicle*'s new velocity. Within the [Merge Platoon] team operator, the Simulate Vehicle operator is 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, which always has an up to date representation of the vehicle it cannot sense when changing lane.
- *Safety Observers:* a role filled by one or more agents. The constraint on the role fillers, is that they must be in a position ahead from the manoeuvre

executor, so they can monitor dangers in advance. Using the communication selectivity presented in Section 4.3.2, agents in the *Safety Observers* role communicate their belief about dangers or unsafe deceleration to others, by taking in account the dangers of sudden movements during the execution of a manoeuvre. Agents filling this role conjointly execute the [Monitor task safety] safety team operator, therefore executing observation plans individually.

4.3.2. Team Framework Operators To ensure a coherent execution of the domain-level operators, STEAM's framework incorporates architecture-level operators (STEAM operators). These operators include: (i) Coherence Preserving (CP); (ii) Monitor and Repair (MR); and (iii) Selective Communication (SC). The CP and MR operators are not presented in this paper, since their use in our application is very limited, so the reader should refer to [15] for more information. In contrast, the CP operator has been very useful in order to support the "intra-team" communications of our CDS, so it is detailed below.

The Selective Communication (SC) operator's task is to synchronize mutual beliefs within the execution of team operators (domain-level operators). A SC operator simply monitors the agent's local beliefs and compares them with the team's mutual beliefs. If it considers that the difference between these two beliefs is important enough, it automatically communicates the agent local belief to other team members and makes its local belief a mutual belief. In order to determine if a new belief should be communicated, the SC operator uses the decision tree represented in Figure 6. According to this tree, the SC operator communicates the new belief considering a reward based on the following variables:

- ρ: the probability that the new belief is not known by its teammates.
- σ: the probability that the new belief opposes a threat to the execution of the current team operator.
- C_c: the cost of communication.
- C_n: the cost of nuisance.
- C_{mt}: the cost for miscoordination.
- S: a reward for synchronization of the team's belief during the execution of a team operator.

By using the decision tree in Figure 6, the SC operator chooses to communicate iff the reward of making a new belief "mutual" is higher than the cost of communications. With a probability of $1 - \sigma$, the new belief is not a threat to the team operator, and therefore, there is no reward relating to this belief. If the agent chooses not to communicate (No COMM (NC)) and the belief is a threat, with a probability $1 - \rho$, this belief was already known by its teammates so the agent receives the standard reward of S. However, with a probability ρ, this belief was not mutual, so the agent receives the standard reward of $S - C_{mt}$, where, in our application, C_{mt} depends on the difference between the local and mutual belief (if the local belief is much different from the mutual belief, C_{mt} is high). Therefore, from Figure 6, the expected utility of not communicating can be defined as $EU(NC) = \sigma * S - (\rho * \sigma * C_{mt})$.

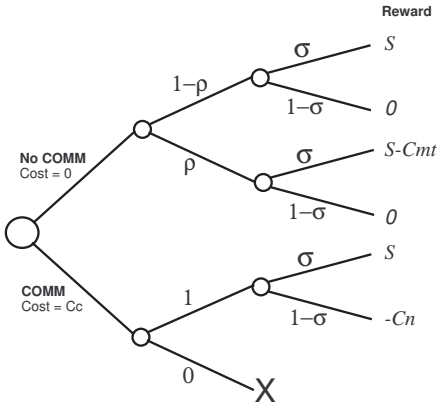

FIGURE 6. Decision tree on team communicative acts, from [15].

In the situation where the agent decides to communicate (COMM (C)), a cost of communication C_c is applied to further possible rewards and this cost is fixed considering the current available communication bandwidth. Following a decision to communicate, the new belief is definitely mutual, so the second branch is irrelevant in Figure 6 (probability 0). However, the new belief can be a threat with a probability σ, in which case the agent receives a reward of S. Nevertheless, with a probability $1-\sigma$, this was not a threat and the communication of this belief has a cost of nuisance C_n to other team members. This cost depends on the type of information that is communicated, but it is usually set to discourage agents from communicating information that may disturb other team members, when it is not necessary. According to this definition, the expected utility of communicating a new belief to team members can be summarized as $EU(C) = \sigma * S - (C_c + (1-\sigma) * C_n)$. The two previous equations on expected utility can be merged to represent the decision of the Selective Communication (SC) operator, which communicates when $EU(C) > EU(NC)$, i.e., iff:

$$\rho * \sigma * C_{mt} > (C_c + (1-\sigma) * C_n)$$

This kind of decision-theoretic selector being part of the STEAM framework, we use it for all of the roles presented earlier. Probabilities, cost and rewards used by the communication decision tree are initialized according to common knowledge on Collaborative Driving System (CDS) and should be adapted through testing using offline learning approach on patterns of communication within the team. For the moment, the SC operators have been very useful to determine when a *Safety Observers* or a *Virtual Vehicle* should communicate its new beliefs. For instance, if the *Virtual Vehicle* (vehicle *F1* in Figure 1) has to modify its velocity during the merge, the probability ρ that this new information on *F1*'s velocity is commonly known mainly depends on the probability P(*L2*,*F1*) that the merging vehicle *L2* has *F1* in its sensor's range (if *L2* is in the platoon). Furthermore, the probability

σ that this information opposes a threat to the merge manoeuvre depends on the difference between *F1*'s local belief on its velocity and the team's mutual belief. If the team is highly out of synchronization, the agent will communicate at a higher probability. Finally, cost C_n will be low for this type of belief, while cost C_{mt} will be higher, since changes on the virtual vehicle's velocity affect the platoon safety.

5. Evaluation and Results

To develop the previous theory on coordination strategies, we have used an agent development toolkit called JACK Intelligent Agents^{TM}[2] which supports the Belief Desire Intention (BDI) agent model [13], as well as teamwork related strategies. The BDI model allows us to develop our application in such a way that each vehicle's driving agent has a database of beliefs and a set of predefined desires that rule its intentions, i.e. change lane, which result in the application of actions like merging and splitting from a platoon. In the Team Oriented Programming (TOP) vision relating to the STEAM model, a non-negligible advantage is the reusability and flexibility of the operators [15], since it contains many infrastructure rules that are not directly related to the domain level. Thus, using JACK's representation of an agent, we managed to develop collaborative driving teams that follow TOP models.

The coordination models presented in Section 4 have been implemented according to the architecture presented in Figure 2. We show as an example in Figure 7's graphic, the results we got in the average coordination of a vehicle exit (split manoeuvre), using a centralized coordination (in the two blue lines) as opposed to the teamwork model of coordination (in the red blue lines). This graphic shows results using the splitter's (vehicle *F2* in Figure 1) data and the splitter's preceding vehicle's (vehicle *F3*) data. The preceding vehicle senses the splitting vehicle and has to adjust from its departure by keeping a *safe* gap until the splitter is safely out. The two bold lines present the difference between, the inter-vehicle distance (IVD) 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 two thinner dotted lines only show the inter-vehicle distance (IVD) from *F3*'s sensor, without applying a difference.

Around time $14s$ in Figure 7, vehicle *F3* has to create a larger and safer distance with the splitting vehicle, so the two bold lines drop, but are readjusted within almost $10s$. The second outlined step arises at time $30s$, and $27s$ 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 IVD curves, but there is no gap considering the distance defined as *safe* (bold lines). Before the splitter has stabilized itself on the next lane (time $37s$ or $34s$), the gap creating vehicle of the centralized model does not manage to keep the *safe* distance and has a difference of $2m$ 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, the splitting vehicle maintains the right *safe* distance more easily in the teamwork model. This approach gives much better results, since the difference with the *safe* distance does not go higher than $0.5m$. When the splitter is stable in the other lane, the distance qualified as *safe* drops to the normal intra-platoon distance. At that moment, the vehicle is at $17m$ (length of the gap created by the vehicle that left) from the *safe* distance, reached by the end of the manoeuvre. 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 manoeuvre.

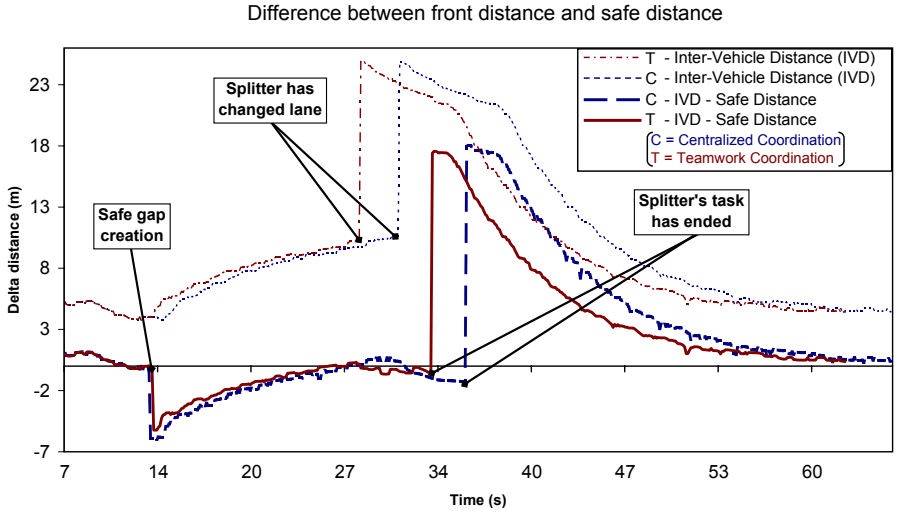

FIGURE 7. Inter-vehicle distance with the splitting vehicle.

The implementation of four different coordination strategies leads us to conclusions concerning their respective advantages and disadvantages. First, as mentioned in Section 4.1, a fully centralized coordination at its simplest version was developed. 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 manoeuvre. The fourth approach uses the previously detailed teamwork model.

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 each manoeuvre. On the second pair of columns, we compare the amount of plans (JACK plans), which were required to support each coordination model, showing

the flexibility of their respective framework. These JACK plans refer to the coordination plans developed in JACK Intelligent Agents^{TM}language[1] that made it possible to support a specific manoeuvre, with a specific communication protocol.

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

Coordination	Nb Messages		Jack Plans	
	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

By using the results presented in Table 1, along with the results presented in Figure 7, each coordination model were analyzed and their respective advantage and disadvantage are summarized as follows:

1. The first centralized coordination 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 coordination 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 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 decentralized coordination model uses less messages, but is a lot less safer than the other models. This fact will be proven using further simulations, but since it uses only two actors, and no virtual vehicle (as mentioned for the teamwork model in 4.3), this model 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 models. It must be mentioned that this number varies more than in the other models, from different contextual simulations, because of the selective communications. On the other hand, the selective

[1]In JACK, Plans are pre-compiled procedures based on the PRS architecture [6]. A plan can only be executed when it answers to internal or external events using *relevance* and belief *context* criterions.

communications enable a faster and safer execution of our manoeuvres, as shown in Figure 7. Using a TOP model implemented with the STEAM vision in JACK, even though there are more actors (to increase safety), 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 their communications.

6. Conclusion and Future Work

Collaborative driving is emerging in the domain of ITS and it will ultimately be part of the every day vehicle's automation system, since it is the next step, following the Adaptive Cruise Control (ACC). CDS can therefore be easily included in the development plans of ITS for the upcoming years, as it will evolve until vehicle level technologies like ACC meet AHS infrastructures technologies and increase ITS benefits on safety, efficiency and environment.

This paper presented an autonomous driving system based on a strong architecture giving a wide latitude to the definition of different inter-vehicle communication models. From these models, we have presented and tested four different strategies, for which advantages and disadvantages were presented. The coordination model based on teamwork presented great avenues considering its flexibility and its ability to safely and efficiently execute manoeuvres.

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 our simulator, similarly to the approach we have taken in [9].

References

[1] M. Antoniotti, A. Deshpande, and A. Girault. Microsimulation analysis of multiple merge junctions under autonomous ahs operation. In *Proocedings of the IEEE Conference on Intelligent Transportation System (ITSC) 97*, pages 147–152, November 1997.

[2] AOS. JACK Intelligent Agents^{TM} 4.1. Software Agents Development Framework, 2004.

[3] Auto21. [Online], April 2004. `http://www.auto21.ca/`, (accessed the 30th of April 2004).

[4] S. Vahdati Bana. Coordinating automated vehicles via communication. Ucb-its-prr-2001-20, University of California, Berkeley, September 2001.

[5] DAMAS-Auto21. [Online], April 2004. `http://www.damas.ift.ulaval.ca/projets/auto21/`, (accessed the 30th of April 2004).

[6] M.P. Georgeff and F.F. Ingrand. Real-time reasoning: The monitoring and control of spacecraft systems. In Computer Society Press, editor, *Proceedings of the Sixth IEEE Conference on Artificial Intelligence Applications(CAIA 90)*, volume 1, pages 198–205, Los Alamitos, Calif., May 1990.

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

[8] Simon Hallé. Automated highway systems: Platoons of vehicles viewed as a multiagent system. Master's thesis, Université Laval, Canada, January 2005.

[9] Simon Hallé and Brahim Chaib-draa. Collaborative driving system based on multiagent modelling and simulations. *Transportation Research Part C: Emerging Technologies*, 2005. Submitted.

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

[11] Petros Ioannou and Margareta 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.

[12] John Lygeros, Datta N. Godbole, and Shankar S. Sastry. Verified hybrid controllers for automated vehicles. *IEEE Transactions on Automatic Control*, 43:522–539, 1998.

[13] A. S. Rao and M. P. Georgeff. Bdi agents: from theory to practice. In *Proceedings of the First International Conference on Multiagent Systems*, pages 312–319, San Francisco, 1995. The MIT Press.

[14] R. R. Stough. *Intelligent Transportation Systems: Cases and Policies*. Edward Elgar Pub. LTD, 2001.

[15] 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.

[16] S. Tsugawa, S. Kato, K. Tokuda, T. Matsui, and H. Fujii. A cooperative driving system with automated vehicles and inter-vehicle communications in demo 2000. In *Proceedings of the 2001 IEEE Intelligent Transportation Systems Conference*, pages 918–923, August 2001.

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

[18] Qing Xu, Karl Hedrick, Raja Sengupta, and Joel VanderWerf. Effects of vehicle-vehicle / roadside-vehicle communication on adaptive cruise controlled highway systems. In *Proceedings of IEEE Vehicular Technology Conference (VTC)*, pages 1249–1253, Vancouver, Canada, September 2002.

Département d'informatique et génie logiciel, Université Laval, Sainte-Foy, QC, Canada, G1K 7P4
E-mail address: `halle@iad.ift.ulaval.ca`

Département d'informatique et génie logiciel, Université Laval, Sainte-Foy, QC, Canada, G1K 7P4
E-mail address: `chaib@iad.ift.ulaval.ca`