CALIFORNIA PATH PROGRAM
INSTITUTE OF TRANSPORTATION STUDIES
UNIVERSITY OF CALIFORNIA, BERKELEY

Coordinating Automated Vehicles via Communication

Soheila Vahdati Bana
University of California, Berkeley

California PATH Research Report
UCB-ITS-PRR-2001-20

This work was performed as part of the California PATH Program of the University of California, in cooperation with the State of California Business, Transportation, and Housing Agency, Department of Transportation; and the United States Department of Transportation, Federal Highway Administration.

Report for MOU 318

September 2001

ISSN 1055-1425

Coordinating Automated Vehicles via Communication

by

Soheila Vahdati Bana

B.A. (University of California at Berkeley) 1990
M.S. (University of California at Berkeley) 1994

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor Pravin P. Varaiya, Chair
Professor Shankar Sastry
Professor Robert Horowitz

2000

The dissertation of Soheila Vahdati Bana is approved:

Chair Date

Date

Date

University of California at Berkeley

2000

Coordinating Automated Vehicles via Communication

Soheila Vahdati Bana

Abstract

Coordinating Automated Vehicles via Communication

by

Soheila Vahdati Bana

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California at Berkeley

Professor Pravin P. Varaiya, Chair

This dissertation addresses the coordination of automated vehicles in an Automated Highway System (AHS). Traffic in an AHS is organized as tightly spaced *platoons* to increase road capacity and safety. Vehicles in AHS are automated and their safe interaction is the subject of this research. This dissertation discusses issues in the design and implementation of a controller for automated vehicles that coordinates the interaction between vehicles. We first define a formalism for safe interaction of automated vehicles and then design a controller algorithm for an individual vehicle that guarantees such safe interaction. The algorithm for the vehicle controller requires real-time information about the relative positions and planned actions of its neighboring vehicles. Therefore, a communication structure is needed to provide this information. We propose a communication structure of local and wide area networks, LANs and WANs, that allows automated vehicles to exchange this information. We also explain communication address resolution in the context of AHS

and propose address resolution schemes for one-lane and multi-lane AHS. The scheme for a multi-lane AHS is space-time division multiple access (STDMA) which is an innovative solution to control multiple user access to the communication channel. Finally, we present a vehicle positioning system by using spread spectrum magnetic signals which is used in STDMA communication and coordination control.

Professor Pravin P. Varaiya
Dissertation Committee Chair

To my family,

whose love and wisdom I enjoy at every step of my life.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my advisor Professor Pravin Varaiya who by trusting my goals and ambitions provided me the opportunity to pursue my education in Communications and Networking. I am grateful for his careful evaluation of my research and meaningful feedback on my dissertation. Pravin not only encouraged my creativity to flow, but also patiently taught me to turn the flow of my thoughts into concise words and concrete expressions. Moreover, I am glad I *learned* to be self-motivated in my research from his unique style of advising and conducting research.

I also thank Professor Shankar Sastry for his sincere encouragement as well as his technical and professional guidance, Professor Roberto Horowitz for valuable advice during my research, and Professor Karl Hedrick for serving on my qualifying exam committee.

I am indebted to Professor Roger Howe for his confidence in my success which at first I borrowed and finally internalized. I thank Professors Costas Spanos and Elijah Polak for their understanding and continuous support.

I am grateful to my friend Mireille Broucke for being my mentor and showing me the roads and backroads to self-confidence and success.

I am glad that in my PhD years I had the pleasure and benefit of interacting with Nick McKeown, Richard Edell, John Koo, Joao Sousa, Angela Chuang, Huma Dar, Farrokh Eskafi, Ekta Singh, Linda Kamas, Angela Wang, John Lygeros, Datta Godbole, Claire Tomlin, John Davis, Ralph Neff, Remco Litjens, Chao Chen, Priya Viswanath, Anuj Puri, and Gaurav Agarwal.

I thank Dr. Luis Alvarez, Megumi Harada, Gabriel Gomes, Charmaine Toy, and

Jingang Yi whose valuable suggestions have contributed to the research that appears in Chapter 2. Chapter 4 of this dissertation has greatly benefited from technical suggestions by Bob Lorenz to whom I am thankful.

My sincere appreciation goes to Dr. Sheila Humphreys who has always been there for me and helped me learn from the wisdom of many other women who were ahead of me, including Paula Hawthorn and Barbara Simons who established the tradition of sharing wisdom in pursuing success for the Women in Computer Sciences and Electrical Engineering (WICSE). And I will remember WICSE for all the supportive smiles and complaints and laughter that we had during our Friday lunches together.

I had the wonderful experience of drafting the Student Parent Policy with Huma Dar, whose company is a blessing, and Caroline Tice. Again, it was Sheila who helped us with the draft and carried it all the way to become a campus wide policy. I would like to extend my gratitude to Professors Randy Katz and Andy Neureuther and the EECS faculty for their sincere support of the students; their approval of the Student Parent Policy allowed me finish my research at my own pace alongside my family. Special thanks are due to Ginger Ogle who set an example for us by initiating departmental policies.

My other wonderful experience in graduate school has been my research on improving the graduate school environment for women with Soha Hassoun. Her enthusiasm was inspiring and I thank her for the joy of the research and presenting the results to students and professors nationwide.

I sincerely thank the wonderful staff and student assistants of the UC Child Care Center whose exceptional care and teaching of my children enabled me to attend graduate

school. I specially thank Yolanda Ross, Dr. Jim Stockinger, Parvin Pourasef, Jill Meador, Christine Hansel, Michelle Harris, Mary Clark, and Diane Wallace.

The sincere, personal and warm approach of the staff of the Office of Graduate Matters has made my graduate experience pleasant and I truly appreciate the caring I have received from Ruth Gjerde, Ruth Tobey, Heather Brown, and Genevieve Thiebaud. I am also thankful to all the staff whose smiles and attention I enjoyed during my pregnancy and afterwards. I am glad to have known Loretta Lutcher.

I am grateful to Ruth Varaiya who helped me get in touch with my body and my mind through her precious Yoga lessons.

Most of all I am thankful to my family, specially my mate, Reza, who understood my career goals, and to my parents who, although themselves never experienced the pleasure of attending school at any level, highly valued education. Finally, I wholeheartedly thank my children who learned to be patient with Mommy and her school.

Chapter 1

Introduction

1.1 Vehicle Automation

Automation is an important focus of current research in transportation systems. The increasing number of cars, trains and airplanes heighten concern about safety, congestion, and the environment. The demand for better transportation systems may be met through vehicle automation. Recent studies that focus on automation of highway systems, air traffic management, and train control predict that vehicle automation will increase safety by reducing human error which is a major contributing factor to accidents. Moreover, studies indicate that automated vehicles will use transportation resources more efficiently and reduce congestion and environmental pollution.

1.1.1 Control and Safety of Automated Vehicles

An automated vehicle should have accurate sensors and precise actuators. The sensors provide information about the surrounding conditions and estimate the dynamic

capabilities of the vehicle. The actuators control the vehicle's dynamics according to a set of pre-designed control parameters. The disturbances that may jeopardize the safety of an automated vehicle can be classified as follows.

1. Exogenous signals such as sensor noise;
2. Poorly modeled or unmodeled dynamics;
3. The actions of other vehicles which can be considered as uncontrollable disturbances when we assume no cooperation among vehicles.

The last source of disturbance, which arises from interaction of automated vehicles, is the subject of our study. A centralized controller that controls the action of every vehicle can eliminate such disturbances by *coordinating* the actions of all vehicles, i.e., computing a set of global optimum actions for vehicles. However, the cost and complexity of a centralized controller may be prohibitive to system designers. We discuss a methodology to design distributed control policies for automated vehicles that help eliminate unsafe and, more generally, undesired interaction of vehicles. Furthermore, we show that inter-vehicle communication protocols can implement the desired control policies. While the principle of our approach can be applied to any type of automated vehicles (air vehicles, trains, etc.), we focus our attention on the automated vehicles in the case of Automated Highway Systems (AHS).

1.2 Automated Vehicles in AHS

Automated Highway Systems (AHS) are proposed as a solution to highway congestion and as a measure to reduce the frequency of accidents and environmental pollution [1, 2, 3, 4, 5]. Traffic in AHS is organized in groups of tightly spaced vehicles called platoons [1, 6] that follow one another closely $(1 - 3m)$. The inter-platoon distance, on the other hand, is large $(> 30m)$. Contiguous vehicles in one lane perform *join* and *split maneuvers*, respectively, to decrease or increase their distances from each other. A vehicle performs lane change maneuver to change lanes. The maneuvers are performed by the automated vehicle through pre-designed control parameters.

AHS uses a hierarchical controller. Figure 1 gives a block diagram of the controller architecture. Starting at the top, the controller layers are called network, link, coordination, and regulation layers [1, 3, 4, 7, 8, 9]. There is one network layer controller for the whole automated network that assigns a route to each vehicle and controls the flow of vehicles in the network. There is one link layer controller for a long segment of each highway that assigns a lane and a speed to vehicles to smooth flow, avoid congestion, and adapt to incidents such as major collisions. There is a coordination layer controller and regulation layer controller for each vehicle. Their respective tasks are to decide on maneuvers that realize the assigned lane, and execute these maneuvers.

1.2.1 Vehicle Controller

The controller layers of an automated vehicle form a hierarchical hybrid controller that consists of a discrete controller (coordination layer) on top of a continuous controller

(regulation layer). The layer that corresponds to coordinating actions of automated vehicles and to reducing the third type of disturbance is the coordination layer. The coordination layer receives optimum route advice from the link layer and issues commands to the regulation layer. Meanwhile, it receives feedback from the regulation layer regarding maneuver completion and provides information regarding traffic flow and density to the link layer.

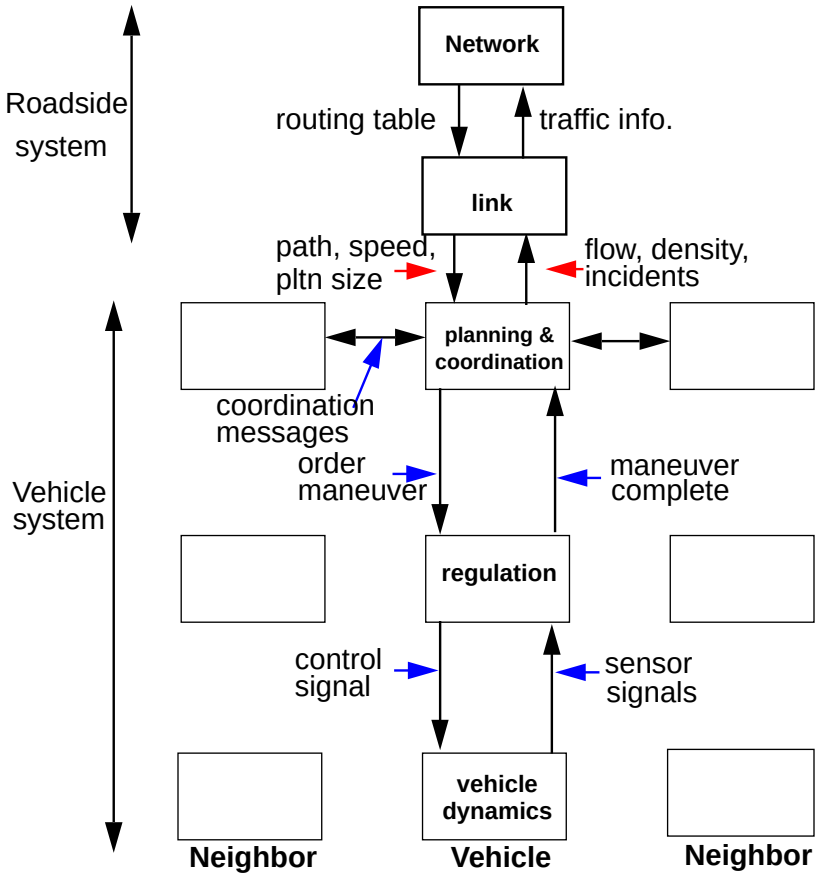

Figure 1.1: *AHS Control Architecture*

1.3 Existing Work

The existing work on the vehicle controller includes designs for coordination and regulation layer controllers, and modeling of the interaction between them as two layers of a hybrid controller.

The current design of the coordination layer controller [10] is a set of point-to-point communication protocols such that for every maneuver only two vehicles communicate with each other without considering the actions of other neighboring vehicles. A crucial point, such as the amount of disturbance that one maneuver may cause to another, has not yet been fully investigated.

The current design of the regulation layer controller [2, 3, 11, 12] provides safety conditions and proves some safety results for vehicle maneuvers and platooning. The regulation layer controller is designed such that the interaction between every two contiguous vehicles in one lane is safe under the following assumptions.

- The weather condition is normal, i.e., no rain, snow, storm, etc.
- Vehicles have relatively uniform masses.
- Vehicles have relatively uniform dynamic capabilities.
- Each vehicle is involved in at most one maneuver at a time.

The above assumptions give rise to questions such as the following: Can vehicle safety be guaranteed when the weather is rainy? What happens when vehicles of different masses collide? Is the third assumption regarding "uniform dynamic capabilities" practically feasible? A lane change maneuver could involve three or more vehicles. How can the last

assumption be justified in case of simultaneous lane change maneuvers when every maneuver communication is only between two vehicles?

These questions in turn give rise to the concern about whether regulation layer safety is sufficient to guarantee the overall AHS safety.

1.4 Thesis Contribution

We assume the hybrid controller model [2, 8] for a vehicle controller and define safety criteria for the coordination layer which has previously not been considered. We propose a communication scheme that allows a vehicle to communicate with its neighbors. In this scheme automated vehicles communicate with each other and coordinate their decisions such that they do not jeopardize the safety of one another.

The coordination controller not only adds another safety measure to the overall AHS design but also extends the conditions under which vehicle safety is guaranteed. In addition, the coordination controller increases AHS efficiency without compromising safety. Finally, coordinating automated vehicles and eliminating the third source of disturbance provides a smooth ride for passengers and enhances their comfort.

1.4.1 Coordination Safety

The coordination layer provides maneuver commands to the regulation layer, and the regulation controller implements a feedback-controlled velocity trajectory with respect to the coordination layer commands [11, 8, 13]. We devise safety rules at the coordination layer such that vehicles do not cause undesired disturbances to each other. We propose a

coordination layer controller with safety constraints in terms of coordinated decisions by vehicles. By enforcing coordination safety, we can guarantee that unsafe vehicle interactions do not occur at the coordination layer. The third source of disturbance to automated vehicle safety is therefore eliminated. Coordination safety relaxes the regulation safety assumptions and extends the conditions (weather, relative mass of vehicles, different dynamic capabilities, etc.) under which AHS is safe.

The design of the coordination controller ensures safety and eliminates the need for verification.

1.4.2 AHS Efficiency

AHS efficiency is defined in terms of the number of vehicles that utilize the road and their travel times [14, 15]. In traffic flow model, a maneuver is represented by the *space* and *time* occupied by the vehicles engaged in that maneuver [14, 15]. The road efficiency increases as the amount of *space* and *time* taken by vehicles decreases. The safety criteria that we define for the coordination layer not only adds to the overall safety of the AHS, but also reduces the *space* margin that is considered for the regulation layer safety. For example, a lane change can end as a join in the new lane when the vehicles coordinate their maneuvering decisions. Moreover, automated vehicles will not waste *time* on counteracting the disturbances. Finally, coordinated vehicles could maintain a higher speed without jeopardizing safety. Therefore, AHS efficiency increases as the amount of *space* and *time* taken by vehicles decreases, without comprising safety.

1.4.3 Passenger Comfort

The amount of *time* taken by vehicle maneuvers will not be spent on counteracting the disturbances and, thus, vehicles will utilize time more efficiently for their maneuvers. Extra *time* ensures added rider comfort during a maneuver [3, 12] and the time can be efficiently used to increase comfort of passengers.

1.4.4 Implementation Issues

Coordination controller design requires communication among neighboring vehicles in order to verify coordination safety conditions. We propose a communication structure among neighboring vehicles to implement the coordination controller. Communication between vehicles requires that their communication addresses be known to each other. To date, an underlying assumption in AHS related research [1, 8, 9, 10] has been that a vehicle have the communication addresses of vehicles that it needs to contact. We justify this essential assumption by solving the problem of address resolution for automated vehicles.

1.5 Thesis Outline

The dissertation is organized into five chapters.

In Chapter 2 we devise a set of safety constraints for the coordination layer, formulate the coordination control problem, and design a coordination layer controller.

In Chapter 3 we discuss the implementation issues and propose a communication structure. Furthermore, we justify our communication design structure by proposing address resolution schemes for automated vehicles in AHS.

In Chapter 4 we focus on implementation requirements. We propose a local positioning technique for automated vehicles that is required for our proposed communication structure in AHS.

In Chapter 5 we conclude our discussion with an overview about how our work can be extended.

Chapter 2

Coordination Controller: Modeling and Design

Our goal is to design a controller for automated vehicles in AHS so that they can interact safely. An automated vehicle has a hybrid controller consisting of both discrete and continuous controllers [8, 9, 16] that interface with each other [8, 10]. The discrete layer is called the coordination layer in AHS design and is described below. The continuous layer is called the regulation layer and provides throttle, braking, and steering actuator inputs. The interface between the two layers is described in [10, 8]. We assume the current design of the the regulation layer that is given in [8, 11, 12, 16], and design a coordination controller.

Our method to design a controller for the coordination layer is the following. First, with respect to the hybrid controller model we define the coordination *safety constraints*. Second, we focus on defining the *control objectives*. Third, we propose *prioritizing objectives* of different vehicles for temporal and spatial coordination of their interaction. Finally, we

design an algorithm for the coordination controller and prove that the algorithm achieves the coordination objectives while satisfying the safety constraints.

Automated Vehicle Hybrid Controller

An automated vehicle can be modeled as a hybrid dynamic system [16]. The state of a vehicle can be described by a set of discrete and continuous variables. Figure

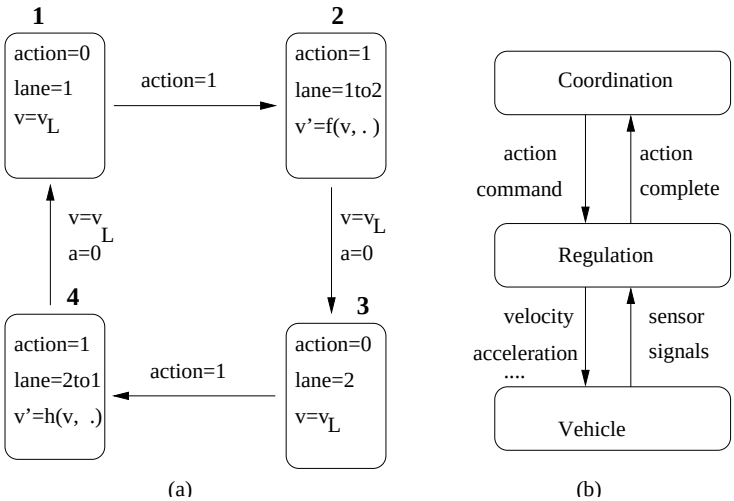

Figure 2.1: *(a) Hybrid controller of an automated vehicle in a two lane automated highway, (b) Hierarchical architecture of the hybrid controller.*

2.1 provides an example of the hybrid controller of an automated vehicle in a two-lane automated highway and depicts its hierarchical architecture. In this example, the discrete state of a vehicle is described by its lane position and continuous states are described by position d and velocity v. When a vehicle is in state 1 (or 3), it is moving along the road and its velocity is constant. A discrete event or *action* command by the coordination layer causes the vehicle to change its state from 1 to 2 (or from 3 to 4) where velocity is changing by the regulation controller. The guard for moving from state 2 to 3 (or from state 4 to

1) is that the velocity has constant values v_L. We are interested in designing a discrete controller that facilitates safe interaction of vehicles by coordinating their discrete events.

One proposed design for AHS assumes "platoons" of vehicles [6, 1, 8, 11]. A platoon is a group of tightly spaced vehicles (2-6 meters). Platoons are separated from one another by large gaps (30-60 meters) [8, 1]. The number of vehicles in a platoon could range from 1 to 10. When platooning is allowed, additional discrete variables such as *platoon-size* and *position-in-platoon* are needed to describe the state of a vehicle. Consequently, more states are generated in the hybrid controller model. We provide a high level design of the coordination controller and we present a method to coordinate discrete controllers of different vehicles. In our design the number of states is irrelevant and our discussion will not include precise details of the vehicle states. The following notation is used in this chapter. We describe later how the parameters that pertain to vehicles change as vehicles take actions.

Symbol	Interpretation
L	a road link
n	number of current vehicles on the link
d^{x1}_{safety}	intra-platoon longitudinal safety margin
d^{x2}_{safety}	inter-platoon longitudinal safety margin
d^{y}_{safety}	lateral safety margin
$\lambda(t)$	space reserved for action λ at time i
Λ	average space taken by a vehicle
p^i	the platoon that includes vehicle i

R^i	configuration of vehicle i
$\mathcal{N}^i$	the neighborhood of vehicle i
u^i	action of vehicle i
U	set of possible actions for a vehicle
$\hat{U}^i$	set of *safe* actions for vehicle i
$\mathcal{N}_c^i(u^i)$	co-maneuvering neighbors of i for action u^i
$\overline{u}_i$	coordinated action of i and its co-maneuvering neighbors
$\overline{u}$	vector of all vehicle actions $(u^1, \ldots, u^n)$
f	function that maps configuration and actions of the neighbors to safe actions
$N_{Left-to-Right}$	percentage of vehicles that must move to the right
$N_{Right-to-Left}$	percentage of vehicles that must move to the left
s^i	current state of vehicle i
$s^i_{\overline{u}}$	the state of vehicle i after the set of safe actions $\overline{u} = (u^1, \ldots, u^n)$
$s^i_{\overline{u}_i}$	the state of vehicle i after any safe set of actions $\overline{u} = (u^1, \ldots, u^n)$ that includes the coordinated action $\overline{u}_i$
C^i	cost of the current state of vehicle i
$C^i_{\overline{u}}$	cost of the state of vehicle i after the set of safe actions $\overline{u} = (u^1, \ldots, u^n)$
$C^i_{\overline{u}_i}$	cost of the state of vehicle i after any safe set of actions $\overline{u} = (u^1, \ldots, u^n)$ that includes the coordinated action $\overline{u}_i$

2.1 Coordination Safety Constraints

Vehicle safety can be defined as maintaining a non-negative distance from other vehicles at all times. Vehicle safety considerations impose constraints on the interaction of vehicles at both the coordination and regulation layers. In this section we define the safety constraints for the coordination controller. First, we consider a spatial safety margin for a vehicle and model vehicle interaction with respect to the safety margin. Then we use this model to design a set of coordination safety rules that allow vehicles to change their relative positions and velocities while maintaining the spatial safety margins.

2.1.1 Safety Margins

Recall that we define coordination *safety* as absence of collision between vehicles as above. Let us define a road *link* L as the road segment between two entry/exit points. Assume that for vehicles on a *link* L, the cruising velocity is determined by the link controller to be the vector v_L where the vector direction is tangent to the road and its magnitude is a function of time. Assuming that initially vehicles in all lanes maintain velocity v_L and are positive distant apart, a collision may happen when one vehicle deviates from v_L. When a vehicle velocity deviates from v_L, other vehicles around it must be able to sense the deviation and adjust their own velocities properly to avoid collisions. We define a *spatial safety margin* d_{safety} as the minimum space that a vehicle i needs to maintain with respect to another vehicle j such that if j has a sudden change of velocity, vehicle i can adjust its velocity appropriately before a collision occurs. The spatial safety margin depends on the relative position of two vehicles as well as on their relative velocities. With respect to the

relative position of two vehicles, the safety margin can be decomposed into *longitudinal* and *lateral* safety margins, d^x_{safety} and d^y_{safety}, respectively, where the x coordinate is along the road and y is perpendicular to x.

The lateral velocity of a vehicle does not change suddenly, but it changes as pre-planned when the vehicle is changing lanes while the longitudinal velocity could have sudden unpredictable changes. This is because when a vehicle realizes that its path may be blocked, it will attempt to stop rather than changing lanes. Thus, the lateral safety margin is small comparing with the longitudinal safety margin. The lateral velocity of a vehicle is zero when the vehicle is cruising and its maximum value, which is realized when a vehicle changes lanes, is relatively small (< 4 meter/second) [8]. When vehicles move along the road, every vehicle is assumed to track the center of its lane. This is a sufficient condition for lateral safety. Therefore, the lateral safety margin can be defined as the lateral distance between two vehicles in two adjacent lanes when they are at the center of their lanes as depicted in Figure 2.2.

In order to determine the longitudinal safety margin, we consider the worst case in which a sudden change in the relative velocity of two vehicles is caused by a sudden

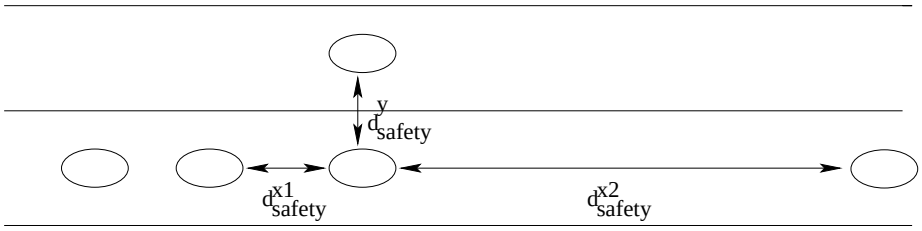

Figure 2.2: *Lateral and longitudinal safety margins.*

deceleration of the front vehicle. For two vehicles in the same platoon, the relative velocity cannot change suddenly because the rear vehicle tracks the velocity of the front vehicle closely through sensors and communication and maintains the same velocity as that of the front vehicle [3]. Therefore, a small safety margin d^{x1}_{safety} on the order of 2-6 meters is sufficient for vehicles in the same platoon. The situation is different when consecutive vehicles are in different platoons. The lead vehicle in the rear platoon does not track the velocity of the tail vehicle in the front platoon. Thus, the relative velocity of the two vehicles may be large and hence, the inter-platoon safety margin d^{x2}_{safety} is between 30-60 meters where we assume that v_L is between $35 - 40$ m/s and maximum deceleration is $4m/s^2$ [8, 11].

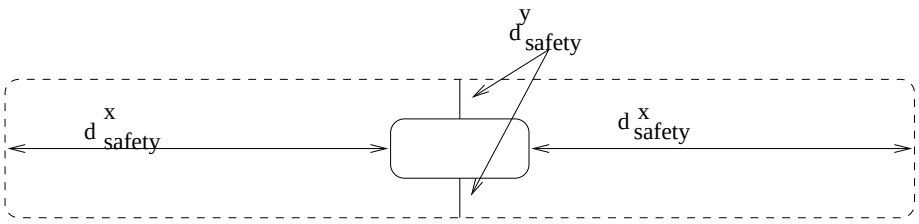

Figure 2.3: *The vehicle safety cell is defined by the lateral and longitudinal safety margins and is moving along with the vehicle.*

The lateral and longitudinal safety margins must be maintained as the vehicle moves and can be considered as a *moving* safety space around the vehicle which we call a *vehicle safety cell*. Figure 2.3 illustrates the concept of the moving safety cell around a vehicle. We define a *platoon safety cell* as the union of the safety cells of the vehicles in the platoon, as shown in Figure 2.4.

Let us define the *interaction range* as the maximal longitudinal range between two automated vehicles whose direct interaction must be controlled for safety purposes. A

vehicle must be able to coordinate its velocity with the vehicles within its interaction range in its lane and the adjacent lanes.

2.1.2 Vehicle Configuration and Neighborhood

A platoon can be represented by a vector p whose components correspond to the vehicles in the platoon. Assume that on the road "front" is defined by the direction of movement along x-axis. We order the position of vehicles in a platoon from front to back, $1, \ldots, n$, where $n = |p|$ is the size of the platoon and the order of the vehicle positions in the platoon determines the order of the corresponding components of p. The platoon to which a vehicle i belongs is denoted by $p(i)$ and the notation $\hat{p} = \emptyset$ is reserved for "empty-platoon". If $|p(i)| = 1$, i is called *Single*. (In [6, 1] a one-vehicle platoon is called a *free agent*.) If i has the position 1 or $|p(i)|$, it is called a *Leader* or a *Tail*, respectively. Otherwise, it is called a *Follower*.

We define the *configuration* of a vehicle i as a vector R^i with components defined below. Each component of R^i represents a platoon within the interaction range of i which is either in the lane of i or in an adjacent lane. The exact positions of the corresponding

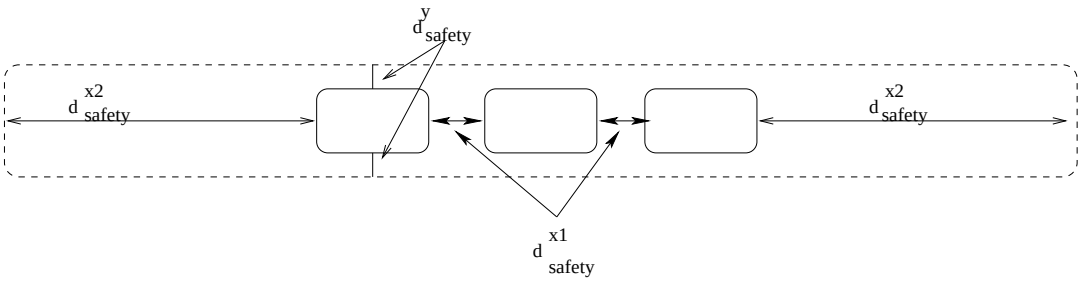

Figure 2.4: *Example of a platoon safety cell.*

platoons may be important to the regulation layer in which case they will be resolved through accurate sensors. However, the approximate relative positions with respect to the safety cell are important to the coordination layer. The components of R^i are ordered in a specific way such that the index of a component in the vector reflects the relative position of its corresponding platoons with respect to $p(i)$ safety cell as explained below and illustrated in Figure 2.5.

$$R^i = [p(i), Front, Behind, R1, R2, R3, L1, L2, L3, RightFront, LeftFront, RightBehind, LeftBehind]$$

The R^i components refer to the platoons in *relative positions* with respect to the safety cell

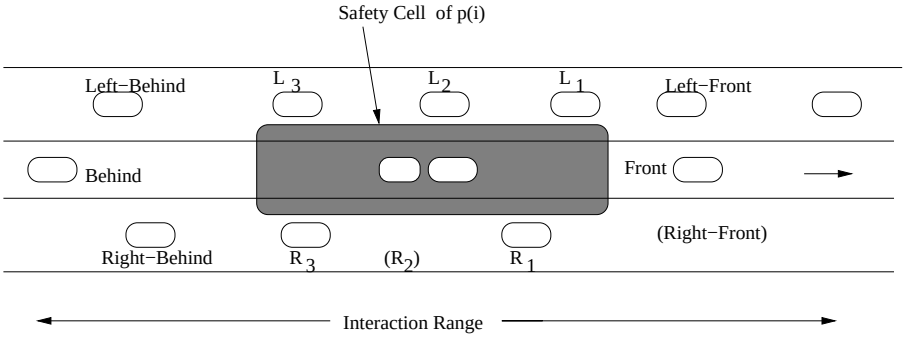

Figure 2.5: *Configuration of a vehicle represents the nearest platoons within its interaction range. In this diagram, the "empty platoon" components of configuration are shown in parentheses.*

of $p(i)$. However, a platoon may or may not exist in a specific relative position with respect to $p(i)$ safety cell and, thus, any component other than the first one could be empty, i.e., $\hat{p}$. The first component represents $p(i)$. The second and third components represent the nearest platoons in front of, and behind, $p(i)$, respectively. $R1$, $R2$, and $R3$ represent the platoons whose leaders are on the right side of $p(i)$ safety cell as shown in Figure 2.5. If the maximum size of any platoon is 10 (vehicles), maximum vehicle size is 5 meters, $d^{x1}_{safety} = 1m$, and

$d^{x2}_{safety} = 60m$, the length of the safety cell of $p(i)$ is 179 meters and there can be at most three platoons on the right side of $p(i)$ that are specified by $R1$, $R2$, and $R3$ from front to back. If there is only one platoon on the right side of $p(i)$, then $R2$ and $R3$ are empty, i.e., $\hat{p}$. Again, this is because only the relative position of platoons is important at the coordination layer and given a specific relative position, the exact position of a platoon will be resolved at the regulation layer. Similarly, $L1$, $L2$, and $L3$ represent the platoons whose leaders are in the left side of $p(i)$ safety cell. The component denoted by $RightFront$ represents the nearest platoon in the right-front position of $p(i)$ as indicated in the figure. Similarly, the components denoted by $LeftFront$, $RightBehind$, and $LeftBehind$ represent the nearest platoons in left-front, right-behind, and left-behind positions with respect to $p(i)$ safety cell as depicted in Figure 2.5. Thus, R^i contains 13 components, some of which may correspond to empty platoons.

We define the *neighborhood* of a vehicle i to be the set of the vehicles in the platoons in R^i. The neighborhood of i is denoted by $\mathcal{N}^i$, and any vehicle in $\mathcal{N}^i$ is called a *neighbor* of i. Based on our definition, $i \in \mathcal{N}^i$ and $i \in \mathcal{N}^j, \forall j \in \mathcal{N}^i$. So the neighborhood relation $\{(i,j)|j \in \mathcal{N}^i\}$ is symmetric. It is not transitive, however. Also, $\mathcal{N}^i \neq \mathcal{N}^j$ if $i \neq j$.

2.1.3 Coordination Actions

A vehicle must coordinate any change in its velocity with its neighbors to avoid unsafe interactions. In the platooning scheme of AHS, vehicles are *cruising* at v_L and a vehicle velocity changes when it wants to become a part of a platoon, or separate from its platoon, or change lanes. We define a *vehicle action* as a discrete event at the coordination layer that commands a vehicle to change its velocity for a finite period of time. The

resulting trajectory of the vehicle depends on the type of action and is carried out through the regulation layer. The regulation controller chooses a feedback-controlled trajectory with respect to the action command, where the feedback is maintained through sensors [2, 7, 8, 11, 12]. When the regulation layer carries out a command from the coordination layer, we say the regulation layer performs a *maneuver* [1, 9]. When a leader takes an action, follower vehicles in the platoon will adjust their own velocities according to the action of the leader.

Join, *split*, and *lane-change* are examples of vehicle actions. In *join* and *split* actions vehicles remain in the same lane and only the magnitude of the velocity changes, while *lane-change* involves changes in the direction of the velocity as well. A *join* is the action that commands the regulation layer to take a trajectory that reduces the safety margin with respect to the front platoon from d^{x2}_{safety} to d^{x1}_{safety}. *Join* action causes a vehicle to join its front platoon. When a vehicle trajectory increases its safety margin with respect to its front vehicle from d^{x1}_{safety} to d^{x2}_{safety}, it is said to have made an *split* action. A *split* action causes a vehicle to split off from its current platoon and become a leader itself. The vehicles that are behind the splitting vehicle in the original platoon will imitate its velocity change and become its followers. A *lane-change* action by a vehicle means that the vehicle moves to an adjacent lane. The regulation layer design provides feedback control only for a single vehicle lane-change [11, 8].

Coordination safety requires that vehicles and platoons maintain their safety cells when they are cruising. When the distance between vehicles is larger than d^{x1}_{safety} but smaller than d^{x2}_{safety}, as in the case of *join* or *split* maneuvers, vehicles must coordinate

their trajectories to avoid collisions. When a vehicle changes lanes, there must be sufficient space in the next lane, which we call a *target cell*, to ensure that there is no lateral collision and that there will be a safety cell for the vehicle after its move. The target cell moves at v_L and has the dimensions of a vehicle safety cell. The midpoint of the vehicle and its target cell must coincide. If vehicle i is changing lanes, coordination safety must guarantee that no other vehicle will move into that moving target cell as i makes its move. Suppose a vehicle i commands an action. We use the term *co-maneuvering* for the vehicles that must coordinate their actions with the action of i to guarantee the safety of the action by vehicle i and denote them by $\mathcal{N}_c^i(m)$ where m denotes the action type and $\mathcal{N}_c^i(m) \subset \mathcal{N}^i$. In Figure 2.6, when vehicle j takes the action join, $k \in \mathcal{N}_c^j(join)$ but $i \notin \mathcal{N}_c^j(join)$. However, when j is taking a split action, then both $i, k \in \mathcal{N}_c^j(split)$.

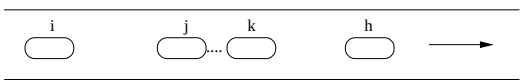

Figure 2.6: *When the action of vehicle j is split, vehicles i and k have co-maneuver actions while vehicle h has cruise action.*

Definition 1 *Let i be a vehicle with action $u^i \neq cruise$ and with co-maneuvering neighbors $\mathcal{N}_c^i(u^i)$ with co-maneuvering actions u^j. The collection of the actions of these vehicles, $\{u^i\} \cup \{u^j : j \in \mathcal{N}_c^i(u^i)\}$, is called a* coordinated action *and is denoted by $\overline{u}_i$.*

When a vehicle i commands an action, its co-maneuvering neighbors must coordinate their actions with the action of i to ensure safety. A co-maneuvering neighbor of i commands its regulation layer to coordinate its action with the action of i through feedback control.

We define the set of *actions*, U, as the following where we include *cruise* as the default action that commands no velocity change of v_L.

$$U = \{cruise, split, join, lane\_change, follow\_join, follow\_split,$$
$$accommodate\_split, accommodate\_join, accommodate\_lane\_change\_side\}$$

where *side* could be *"left", "left-front", "left-behind", "right", "right-front", or "right-behind"*.

The action taken by vehicle i is denoted by u^i, for some $u^i \in U$.

Figure 2.7 provides an example of co-maneuvering neighbors and a coordinated action. The vehicle X can move to the right lane if C and D provide enough space for its target cell. Thus, the set of actions u^X =*lane-change*, u^C =*accommodate-lane-change-left-front*, and u^D =*accommodate-lane-change-left-behind* make a coordinated action, where C, D are co-maneuvering neighbors of X. If C and D do not cooperate with X, then a lane-change to the right by X is not guaranteed to be safe. When a vehicle j is changing lanes, the vehicles in the target lane must coordinate their actions with the *lane-change* action of j. We call the coordinating actions of those vehicles *accomodate-lane-change-side* where *side* could be *"left", "left-front", "left-behind", "right", "right-front", or "right-behind"* depending on the position of the co-maneuvering vehicle in the target lane.

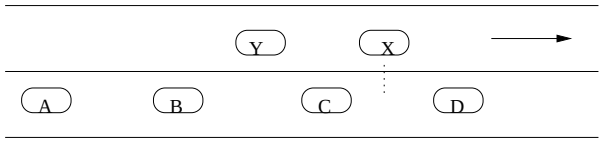

Figure 2.7: *Vehicle X wants to move to the right lane. Vehicles C and D are its co-maneuvering vehicles that must provide a target cell for X and not move into that target cell during the change-lane maneuver by X.*

When a platoon *Leader* takes a *join* action, its *Follower* vehicles will be among its co-maneuvering vehicles. Figure 2.8 depicts another example of the actions of co-maneuvering neighbors. Vehicle j is a $Follower$ in the platoon that is lead by vehicle k. When j is joining k, the vehicles in its platoon that are behind it take a $follow\text{-}join$ action. The platoon leader k and its *Follower* vehicles take an *accomodate-join* action. If j was splitting from its platoon, led by k, its action would be *split*, the action of its followers would be $follow\text{-}split$, and the action of k and its followers would be *accomodate-split*.

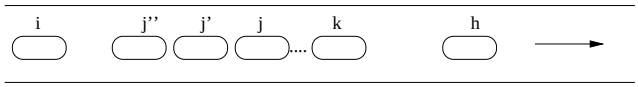

Figure 2.8: *When the action of vehicle j is join, the action of both j' and j'' is follow-join, and the action of k is accomodate-join.*

We formalize below the coordination safety rules with respect to our safety model and the current design of the regulation layer, i.e., the feedback-controlled trajectories that are provided [11, 13, 8, 16].

- There can be only one vehicle or one platoon moving into or out of a platoon safety cell at a time.
- A lateral move for lane-change requires an empty target-cell.
- There can be only one vehicle moving into a target-cell at a time.
- The target-cell for a vehicle must be adjacent to the current safety-cell of the vehicle.

Once these coordination safety rules are implemented, the regulation layer design can be revised with respect to the safety constraints and the longitudinal safety margin for vehicles

could be reduced.

2.1.4 Safety Constraint

Each action of a vehicle at the coordination layer provides a command to the regulation layer. An action u^i is called *unsafe* if it commands a trajectory that causes vehicle i to collide with another vehicle. An action u^i is called *safe* if it is not unsafe.

Coordination safety can be implemented through coordinating vehicle actions. This will result in coordinated motions at the regulation layer and ensure vehicle safety. Assuming all vehicles are initially cruising, they will coordinate their actions with respect to their relative positions. When a vehicle i commands an action, its co-maneuvering neighbors decide on their co-maneuvering actions according to (i) the action of i, and (ii) their positions relative to i. In general, an action considered by itself cannot be categorized as "safe" or "unsafe". Whether an action of a vehicle i is safe depends on the actions of neighbors of i, and their relative positions with respect to i. The actions of the neighbors of i not including i's action are denoted by $\{u^j\}_{j\in\mathcal{N}^i\setminus\{i\}}$, and the relative position of the neighbors is represented by the configuration of i, R^i. We can study each feedback controlled trajectory in combination with different neighborhood configurations and the possible feedback controlled motions of the neighbors in each configuration. In this way we can investigate the possibilities of collisions and determine whether an action is safe or not with respect to its neighbors configuration and their actions. Consequently, we can devise a set of *coordination safety rules* that defines a safe action of i with respect to the actions of its neighbors and their relative positions. That is, we can define a map f such that given (i) the action of the neighbors of i, and (ii) the relative positions of i and its neighbors, f provides a

set of safe actions by i. The function f is provided by the regulation layer. For different configurations, the regulation layer designers determine the feasibility of safe coordinating actions with respect to relative positions of vehicles and their dynamic capabilities. The actual distances of vehicles are not crucial at the coordination layer; however, they determine the exact vehicle trajectories at the regulation layer. The distances between vehicles are measured (by radars) at the regulation layer and although the type of the vehicle trajectory is determined according to the type of action, the exact trajectory is set by the distance. Thus, the regulation layer designers provide trajectories for the coordination layer actions, i.e, define the set of actions for the coordination layer. Thus, given the actions of neighbors of a vehicle and their relative positions, the map f depends on whether a velocity trajectory exists for the vehicle to safely coordinate its action with its neighbors or not. If a vehicle is not required to coordinate its action with any of its neighbors for safety purposes, then it continues to *cruise*. The function f may assign empty set to the pair $(R^i, \{u^j\}_{j\in\mathcal{N}^i\setminus\{i\}})$ if the actions of the neighbors of i are such that a collision cannot be avoided by any of the possible actions of i (including *cruise*).

Therefore, coordination safety rules for a vehicle i can be formulated by regulation layer designers as a function of its configuration and the actions of its neighbors.

$$(R^i, \{u^j\}_{j\in\mathcal{N}^i\setminus\{i\}}) \xrightarrow{f} \hat{U}^i,$$

where $\hat{U}^i \subseteq U$.

The *coordination safety constraint* is defined by the conditions that $\hat{U}^i \neq \emptyset$ and $u^i \in \hat{U}^i$ for every vehicle i, i.e.,

$$\hat{U}^i \neq \emptyset \quad \wedge \quad u^i \in \hat{U}^i, \quad \forall i \tag{2.1}$$

Example 1 *Figure 2.9 illustrates examples of violating the safety constraint.*

Case (a): Assume that j and k are in the left lane of i and are moving to the right lane. At the same time, h is splitting. When j, k and h are moving into the safety cell of i simultaneously, there is no safe action by i and $\hat{U}^i = \emptyset$. This is because cruising is not safe when other vehicles are in the safety cell, and lane-change is not allowed without a clear target cell.

Case (b): When h is splitting from its platoon, a join *action by i is not a safe action,* join $\notin \hat{U}^i$.

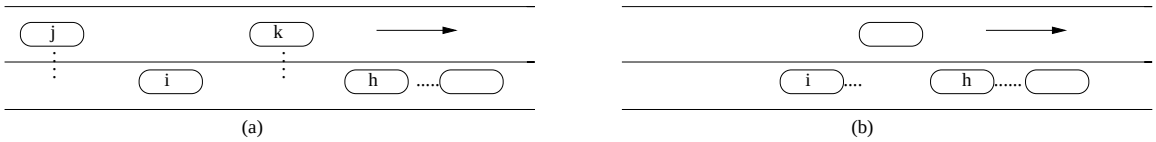

Figure 2.9: *Safety constraint for vehicle i depends on the relative position of its neighbors and their actions as explained in Example 1.*

We summarize the discussion above. Let $\overline{u} = (u^1, \ldots, u^n)$ be a vector of vehicle actions. Then $\overline{u}$ is safe if each vehicle action u^i satisfies the constraint 2.1. In the next section we investigate how to select the best safe action.

2.2 Coordination Control Objectives

The two main considerations in control of vehicle interaction is *vehicle safety* and optimizing *road utilization*. For a given average velocity for vehicles, road utilization increases as the average space taken by each vehicle decreases. The amount of space taken by a vehicle is a function of the actions of the vehicle. We can estimate the average space

reserved by a vehicle as follows. We first calculate *space-time* reserved for an action as described in [14]. Calling $\lambda(t)$ the space reserved for an action at time t, and T the duration of the action, the space-time used by an action is computed as the integral of space with respect to time $\int_0^T \lambda(t)dt$ where the space includes safety margins. The average space utilized by a vehicle, Λ, taking N different actions over a time $T = \sum_{i=1}^{N} T_i$ is

$$\Lambda = \frac{\int_0^{T_1} \lambda_1(t)dt + \int_0^{T_2} \lambda_2(t)dt + \ldots + \int_0^{T_N} \lambda_N(t)dt}{T}.$$

Assuming that the average velocity of a vehicle when it takes different actions is nearly constant, the overall time a vehicle spends on a given road link is constant, too. Thus, the average space that a vehicle takes depends on the space-time taken by its actions. In order to minimize the average space taken by a vehicle it is desired to minimize the number of actions that take a large amount of space-time, or equivalently those actions that require a large amount of space. *Lane-change* is an example of an action that takes a large amount of space because it requires an empty target cell. When there are uncoordinated actions vehicles may need to abort their actions for the purpose of safety and attempt their actions later. This results in wasted space-time by aborted maneuvers and a larger average space per vehicle. Therefore, lack of coordination could reduce the overall road utilization.

The coordination layer of a vehicle is responsible for coordinating the vehicle interaction with other vehicles as they take different actions to form or break platoons, and change lanes. We have not yet mentioned the purpose for the actions: What is the objective of a vehicle when it takes an action? Why should a vehicle join a platoon, or split off a platoon? How does a vehicle know which lane it should be moving in, and when it should change lanes? The objectives for vehicle actions are provided by the link layer. Thus, the

coordination layers of vehicles receive the objectives from the link layer and select actions to implement these objectives.

In this section we first overview the link layer controller and its interface with the coordination layer as described in [8, 11]. We then suggest improvements to the interface that would allow us to design a coordination controller to improve safety and road utilization.

2.2.1 Link Layer: Overview of Existing Work

The link layer receives real-time information about vehicle density and velocity profile on the road. It then determines the entry flow rate to the link L, and commands the *link cruise velocity*, v_L, and the maximum platoon size, P_L. The link layer also determines the required actions by vehicles at every lane. This is done with respect to the vehicle destinations and current traffic conditions on the road such that traffic load is balanced amongst different lanes and vehicles are able to leave the road at their destination exits. Furthermore, vehicles in general should join each other and form platoons up to size P_L. Only when vehicles are near their destination exit they should become *Single*.

Figure 2.10: *The link layer receives information about the current traffic conditions and demands and provides the link velocity, v_L, maximum platoon size, P_L and the number of vehicles at each lane that must change lanes to the right and left, $N_{Right-to-Left}$ and $N_{Left-to-Right}$, respectively.*

2.2.2 Link and Coordination Interface: Overview of Existing Work

In the existing work the interface between the two layers is vaguely described and the coordination objectives are not well defined for individual vehicles [1, 11, 12, 8, 2]. The link layer provides a maximum platoon size P_L and vehicles in general are supposed to join each other and form platoons that do not exceed some determined maximum size. The link layer also informs vehicles of the next exit so that *Exiting* vehicles split from their platoons and prepare to exit. In addition, the link layer orders a percentage of vehicles at each lane to change lanes to the right and left. When the link layer provides aggregated percentage numbers $N_{Left-to-Right}$ and $N_{Right-to-Left}$ as above [11, 17], the individual vehicles attempt uncoordinated actions. If vehicles have interfering actions, they may have to abort their actions which results in an increased space-time utilization by vehicles.

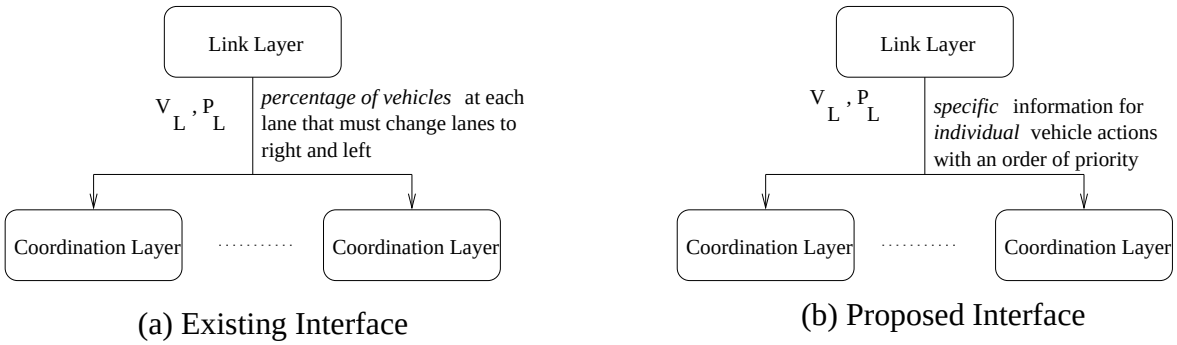

Figure 2.11: *The existing link layer interface does not clarify individual vehicle actions. The proposed design provides sufficient information to vehicles that allows individual vehicles to determine their actions while the order of priority of their actions facilitates their cooperative coordination.*

2.2.3 Link and Coordination Interface: Design Improvement

In this section we propose a new design for the interface between the link and coordination layers. The new interface allows us to design a controller for the coordination layer which is discussed in the next section.

We need to guarantee that vehicles can coordinate their actions for safety reasons as well as for maximization of road utilization, and so that conflicts of the objectives of individual vehicles do not arise. In this section, we assign different *priorities* to different coordination objectives. Automated vehicles will take their actions according to these priorities. We propose a priority order for the link layer commands, and hence, a priority order of objectives for the coordination controllers.

We are concerned with the the link layer interface with the coordination controller of the vehicles that are on the road. Within a multi-lane automated highway there are different *join*, *split* and *lane-change* action commands that need to be ordered in a priority. In [18] a methodology for dealing with multi-agent, multi-objective problems has been introduced that considers a sequential order of priority for objectives, and is adopted by [7]. We, too, adopt the methodology and propose a sequential order of priority for the link layer commands. We consider examples of one-lane and two-lane automated highways to present our methodology in prioritizing the link layer commands.

Assume that the link layer commands for vehicle actions such as lane-change are feasible in the sense that there exists sufficient space-time for all the commanded actions to take place. We divide a link into sections and consider a top priority action for each section. We will shortly discuss the priority order of the other actions in each link section.

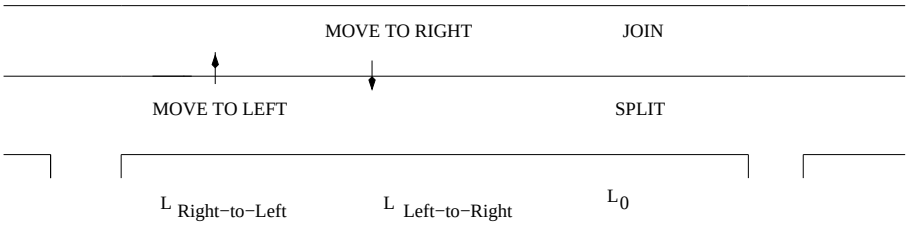

Figure 2.12: *Link division according to the type of actions. At each division of the link a specific action has the highest priority.*

Upon entering the road, some of the vehicles will need to move to the left lane to balance the traffic load. Their moving to the left will open room for *lane-change* actions of exiting vehicles in the left lane that need to move to the right lane. Prior to the exit point, exiting vehicles in the right lane will *split* and prepare to exit while vehicles in the left lane will *join* to maximize road utilization. Figure 2.13 illustrates an example of this temporal ordering of the actions with respect to the link division. Let us divide a link L into sections $L_{Right-to-Left}$, $L_{Left-to-Right}$, and L_0 as depicted in Figure 2.12 where the total length of the sections is equal to the length of the link L. The section $L_{Left-to-Right}$ ($L_{Right-to-Left}$) is where *lane-change* actions to the right (left) have the top priority and L_0 is where no *lane-*

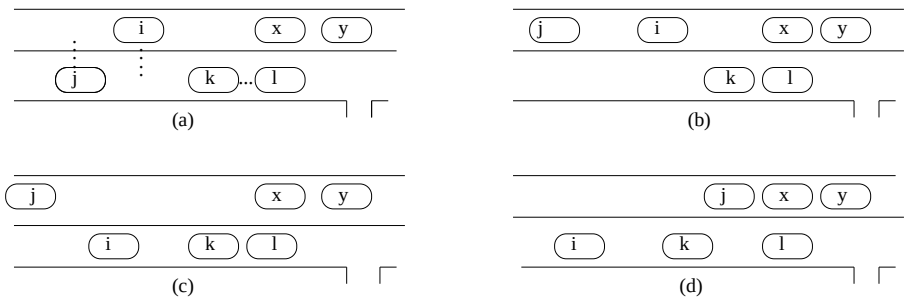

Figure 2.13: *Temporal order of the actions according to the link division: (a) i and j want to change lanes in opposite directions while vehicle k wants to split, (b) j changes lanes, (c) i changes lanes, (d) k splits and prepares to exit while j joins x and y.*

change actions take place. Such division minimizes the amount of interference that may be caused by uncoordinated simultaneous actions. Assuming the *lane-change* actions to left and right require the same amount of space-time, the link layer can estimate $L_{Left-to-Right}$ and $L_{Right-to-Left}$ by the percentage of vehicles that must change their lanes from left to right, $N_{Left-to-Right}$, and from right to left, $N_{Right-to-Left}$, by the following relationship.

$$\frac{L_{Left-to-Right}}{L_{Right-to-Left}} = \frac{N_{Left-to-Right}}{N_{Right-to-Left}}$$

This, however, only specifies the ratio of $L_{Left-to-Right}$ and $L_{Right-to-Left}$. The absolute length of $L_{Left-to-Right}$ and $L_{Right-to-Left}$ is determined with respect to v_L and the amount of space-time that a lane-change maneuver takes and the total number of vehicles on the link. An ideal division of the link will be based on real-time feedback regarding the destinations of vehicles in different lanes and space-time availability.

2.2.4 Prioritizing the Coordination Objectives

We propose a ranking system for the actions commanded by the link layer that will prioritize the commands. This format specifies a "cost" associated with the *state* of each vehicle. In the following we first define the *state* of a vehicle and explain how the link layer assigns cost values to vehicle states. We then explain how a vehicle state, and hence, its cost, change whenever vehicles take actions. The control objective for the coordination layer will then be to minimize the cost through actions. A higher cost associated with a vehicle state reflects its higher priority to take its action.

Let us define the *state* of a vehicle i, s^i, to be specified by *position-in-platoon*, *lane number* and *type* of vehicle i, where these terms are defined below. Suppose the maximum

platoon size is specified by the link layer and is denoted by P_L. The *position-in-platoon* of a vehicle i is an element of the set {*Leader, Tail, Follower, Single, BeyondMax*} where the first three elements specify the position of the vehicle in its platoon, $p(i)$. *Single* refers to a single vehicle i with $|p(i)| = 1$, and $BeyondMax$ indicates that the size of $p(i)$ is beyond the maximum platoon size specified by the link layer, $|p(i)| > P_L$ regardless of the position of i in its platoon. Vehicle *lane number* denotes the current lane of the vehicle. A vehicle *type* is defined with respect to its current location and destination. Namely, if the destination of a vehicle is the next exit, that vehicle is said to be of the $Exiting$ type. Otherwise, it is *non-Exiting*. The type of a vehicle is constant over a link but it changes as the vehicle travels over different links. However, the *lane number* and *position-in-platoon* of a vehicle change by vehicle actions. Thus, the state of a vehicle changes by vehicle actions, too. Recall that in Definition 1 we called the actions of vehicle i and its co-maneuvering neighbors the coordinated action $\overline{u}_i$. As long as the safety constraint for all vehicles is satisfied, the state parameters of a vehicle i are only affected by the coordinated actions that include u^i. This is clear by the fact that a change in the *lane number* of a vehicle requires an action by the vehicle. Similarly, *position-in-platoon* of a vehicle changes when the vehicle takes a *join* or *split* action or coordinates its action with a *join* or *split* action of a neighbor vehicle. More precisely, let vehicle i be in state s^i. Consider a safe vector of actions $\overline{u} = (u^1, u^2, \ldots, u^n)$, i.e., $\forall i, u^i \in \hat{U}^i$. Let the coordinated action $\overline{u}_i$) be the coordinated actions of i and its co-maneuvering neighbors given that i takes action u^i. Let $s^i_{\overline{u}}$ be the new state of vehicle i after the vector of actions $\overline{u}$ have been taken. Then the new state of i depends only on the coordinated action $\overline{u}_i$. In other words, if $\overline{v} = (v^1, v^2, \ldots, v^n)$ is another safe vector of

actions for all vehicles such that $v^i = u^i$ and $v^j = u^j$ for every j that is a co-maneuvering neighbor of i, then $s^i_{\overline{u}} = s^i_{\overline{v}}$, where $s^i_{\overline{v}}$ is the new state of vehicle i after the vector of actions $\overline{v}$ has been taken. Therefore, we can denote the new state of vehicle i after a vector of actions $\overline{u} = (u^1, u^2, \ldots, u^n)$ have been taken as either $s^i_{\overline{u}}$ or $s^i_{\overline{u}_i}$ where $\overline{u}_i$ is the coordinated action of i and its neighbors included in $\overline{u}$. We denote the assigned cost to the current state of a vehicle i by C^i. By the definition of cost, the current cost of i, C^i, changes when the state of i changes. Similarly, the cost of the new state of i can be denoted either by $C^i_{\overline{u}}$ or $C^i_{\overline{u}_k}$.

The link controller determines the desired *lane number* and *position-in-platoon* for every vehicle type with considerations for balancing the traffic load on different lanes. That is, the link layer determines the desired state for each vehicle and orders the vehicles to take actions to realize their desired states. However, the link layer does not directly dictate commands to individual vehicles about the actions they should take. In our proposed interface, the link controller assigns a cost value to every possible vehicle state and it also informs the vehicles of the costs assigned to different states. The assumption is that every vehicle knows its current state. Thus, when provided the cost values for different states, every vehicle figures out its current cost. Moreover, a vehicle learns of the possible states that are associated with lower costs. Hence, the vehicle determines the actions that it could take in order to reach the desired state(s). If there are conflicting objectives among vehicles, the vehicle with the highest cost will take its action.

In our model the link layer does not explicitly command any actions for an individual vehicle, but provides implicit commands by assigning costs to vehicle states. This

is because vehicles execute the actions which most reduce their costs (within safety constraints). The higher a vehicle cost, the higher priority it has to take an action to reduce its cost. In this way, the link layer commands are indirectly carried out. This will be explained in detail below.

Example 2 *The cost of an* Exiting *vehicle when its* position-in-platoon *is* Follower *is high, but its cost when it is* Single *is minimal. Therefore, the vehicle takes appropriate actions to minimize its cost which serves the objective of becoming* Single*, i.e., ready to exit.*

In the next section we rank the link layer commands such that cost minimization by vehicles results in implementing the link layer commands.

2.2.5 Examples of Prioritizing Objectives

In order to further clarify prioritizing objectives through cost assignment, we provide two examples of an ordering by cost for vehicles states. The two examples provide an order by cost of vehicle states in one- and two-lane AHS. The actual value of the cost is not as important, as mentioned above, as the ordering of the cost values. Hence, we provide only the ordering of cost in the following examples.

Example I: Order of Objectives in a One-Lane Automated Road

Let us first consider a one-lane automated road. A vehicle is assumed to know its own state and cost (given by the link layer). Table 2.1 assigns vehicle cost with respect to vehicle state parameters. The following order of the cost values represents our selection of

Cost Values	*Type*	*Platooning Status*
$c_{nE,S}$	non-Exiting	Single
$c_{nE,T}$	non-Exiting	Leader or Tail
$c_{nE,F}$	non-Exiting	Follower
$c_{E,S}$	Exiting	Single
$c_{E,T}$	Exiting	Leader or Tail
$c_{E,F}$	Exiting	Follower
c_{BM}	Exiting or non-Exiting	BeyondMax

Table 2.1: *Cost values for vehicles in a one-lane automated road.*

the priority of actions in a one-lane automated road.

$$c_{E,S} = c_{nE,F} < c_{nE,T} < c_{nE,S} < c_{E,T} < c_{E,F} < c_{BM} \tag{2.2}$$

The above order is justified as follows. First, *Exiting* vehicles have the highest priorities to become *Single*. If an Exiting vehicle is a Follower, it can reduce its cost by first becoming Leader, and then becoming Single. Second, other vehicles must minimize their *space* utilization by forming larger platoons and hence, $c_{nE,F} < c_{nE,T} < c_{nE,S}$. However, the highest priority is to prevent oversized platoons and that is why c_{BM} has the highest value.

Example II: Order of Objectives in a Two-Lane Automated Road

We assume similar cost values for vehicles in a two-lane automated road based on the *type*, *position-in-platoon*, and *lane number* as shown in the following Table. As in the previous example, the order of the cost values determines the order of action priorities in each section of the road link where different lane sections are shown in Figure 3.1.

Cost in Left Lane	*Cost in Right Lane*	*Type*	*Platooning Status*
$c_{L,nE,S}$	$c_{R,nE,S}$	non-Exiting	Single
$c_{L,nE,T}$	$c_{R,nE,T}$	non-Exiting	Leader or Tail
$c_{L,nE,F}$	$c_{R,nE,F}$	non-Exiting	Follower
$c_{L,Ex,S}$	$c_{R,Ex,S}$	Exiting	Single
$c_{L,Ex,L}$	$c_{R,Ex,L}$	Exiting	Leader or Tail
$c_{L,Ex,F}$	$c_{R,Ex,F}$	Exiting	Follower
c_{BM}	c_{BM}	Exiting or non-Exiting	BeyondMax

Table 2.2: *Cost values for vehicles in a two-lane automated road.*

Order of Objectives in First Link Section $\mathbf{L}_{Right-to-Left}$

Suppose that in link section $L_{Right-to-Left}$ the highest priority is *lane-change* from right to left to balance the traffic load. Then, the order of priorities is (i) *lane-change* actions to the left for the *non-Exiting* vehicles, (ii) *join* actions for *non-Exiting* vehicles in both lanes to open space for further lane-change actions, (iii) *lane-change* actions for *Exiting* vehicles in the left lane to the right lane. The objective is to first encourage the *non-Exiting* vehicles to move to the left lane, and *join* platoons, and then allow the *Exiting* vehicles to move to the the right lane. The desirable *position-in-platoon* for *Exiting* vehicles is *Single* and for *non-Exiting* vehicles is *Follower*. We disallow any oversized platoons by assigning highest cost to BeyondMax. By the regulation layer design, a vehicle may change its lane only when it is *Single*. For *non-Exiting* vehicles in the right lane, we set the following order to allow vehicles to become *Single* and be able to move to the left lane.

$$c_{R,nE,S} < c_{R,nE,T} < c_{R,nE,F}.$$

The order of cost values for *non-Exiting* vehicles in the left lane is to reduce *space* utilization by encouraging vehicles to *join*:

$$c_{L,nE,F} < c_{L,nE,T} < c_{L,nE,S}.$$

The following priority holds for *lane-change* actions for Exiting vehicles:

$$c_{L,E,S} < c_{L,E,T} < c_{L,E,F}.$$

We assign a similar order for the *Exiting* vehicles in the right lane:

$$c_{R,E,S} < c_{R,E,T} < c_{R,E,F}.$$

Therefore, the cost values can be ordered as the following where the minimum cost values are $c_{L,nE,F}$ and $c_{R,E,S}$, and the maximum cost value is c_{BM}:

$$c_{R,E,T} < c_{R,E,F} < c_{L,E,S} < c_{L,E,T} < c_{L,E,F} < c_{L,nE,T} < c_{L,nE,S} < c_{R,nE,S} < c_{R,nE,T} < c_{R,nE,F}$$

Order of Objectives in Second Link Section $\mathbf{L}_{Left-to-Right}$

In link section $L_{Left-to-Right}$, the sequence of priorities we wish to implement could be as the following: (i) *lane-change* actions to the right for *Exiting* vehicles, (ii) *lane-change* actions to the left for *non-Exiting* vehicles, (iii) *join* for *non-Exiting* vehicles in both lanes for opening room for incoming vehicles and increasing the road utilization. The objective is for *Exiting* vehicles to prepare to exit by moving to the right lane and becoming *Single*:

$$c_{L,E,S} < c_{L,E,T} < c_{L,E,F}.$$

While the *non-Exiting* vehicles have a higher cost in the right lane, the *non-Exiting* vehicles in both lanes are encouraged to *join* and open *space* for lane change actions by exiting

vehicles:

$$c_{L,nE,F} < c_{L,nE,T} < c_{L,nE,S} < c_{R,nE,F} < c_{R,nE,T} < c_{R,nE,S}.$$

The *Exiting* vehicles must become *Single* in preparation to exit:

$$c_{R,E,S} < c_{R,E,T} < c_{R,E,F}.$$

We combine the above cost orders into the following where, again, the minimum cost values are $c_{L,nE,F}$ and $c_{R,E,S}$, and the maximum cost value is c_{BM}:

$$c_{L,nE,T} < c_{L,nE,S} < c_{R,nE,F} < c_{R,nE,T} < c_{R,nE,S} < c_{L,E,S} < c_{L,E,T} < c_{L,E,F} < c_{R,E,T} < c_{R,E,F}.$$

Order of Objectives in Third Link Section $\mathbf{L_0}$

In the last section of the link, we want to make sure that *Exiting* vehicles will exit, and *non-Exiting* vehicles join to form platoons and provide *space* for entering vehicles. Thus, we continue the same order of cost values as above:

$$c_{L,nE,T} < c_{L,nE,S} < c_{R,nE,F} < c_{R,nE,T} < c_{R,nE,S} < c_{L,E,S} < c_{L,E,T} < c_{L,E,F} < c_{R,E,T} < c_{R,E,F}.$$

This cost ordering scheme could also provide a heuristic performance measurement "metric" for the coordination controller. The smaller the vehicles costs are at the end of the link, the better job the coordination controllers have done. The real-time feedback that the link layer receives could include the sum or the average cost of vehicles. The link layer then can evaluate the extent to which its commands have been implemented.

2.3 Coordination Controller Design

Our goal is to design a controller for the coordination layer of each vehicle that implements the link layer commands. The link layer does not directly assign actions to individual vehicles. Instead, it assigns costs to all vehicle states as in the examples above. These costs are used by the coordination controller to implement the link layer commands.

In this section we first formalize some assumptions about action safety at the coordination layer. Next, we give a brief overview of the communication scheme that is essential in implementing the coordination controller, and then describe the coordination controller design. We then discuss the coordination controller, which is an algorithm that is run by all vehicles to determine safe simultaneous actions. Finally, we prove that the algorithm does find actions that reduce the cost of vehicles safely.

Assumption 1 *Vehicles start within safety margins and cruising at the velocity* v_L.

2.3.1 Communication

Communication plays an important role in our controller design. We propose a communication structure that allows a vehicle to communicate with its neighbors. Recall that neighbors for a vehicle i include all the vehicles with which i must coordinate its action for safety purposes. Communication provides every vehicle current information about the relative positions and states of its neighbors. The safety of a vehicle, as discussed in Section 2.1.4, depends on the actions of its neighbors. Therefore, a vehicle must communicate with its neighbors and coordinate its actions with them for the purpose of safety.

we present a high level model for the communication structure and ignore the

channel behavior.

Assumption 2 *The communication channels are ideal and bidirectional between any vehicle i and its neighbors $j \in \mathcal{N}^i$. Moreover, the exchange of a finite number of messages is instantaneous, i.e., communication time is negligible.*

For the purpose of coordination control, vehicles communicate prior to taking actions. Figure 2.14 depicts a rough schematic of the communication steps that implement the controller algorithm for the coordination layer. During the *communication* mode of each *round*, vehicles run an algorithm together to determine safe and cost reducing actions for each vehicle, where the highest priority actions are determined at the first *step* of communication and further safe and cost reducing actions are determined at the following steps according to their priorities. Vehicles take the actions, as determined in the *communication* mode, in the *action* mode. During the *action* mode, vehicles communicate at the regulation layer which is outside the scope of our discussion. After taking the actions, they begin the next round, i.e., they communicate again at the coordination layer to determine further actions. The rounds of *communication* and *action* continue until no cost can be further reduced.

In the following sections, we discuss the details of the algorithm procedure introduced above for the communication period prior to taking actions. The communication allows vehicles to exchange information regarding their possible actions, negotiate action priorities, and verify the safety of their actions. The communication requirement for the coordination layer and related implementation issues are discussed in later chapters.

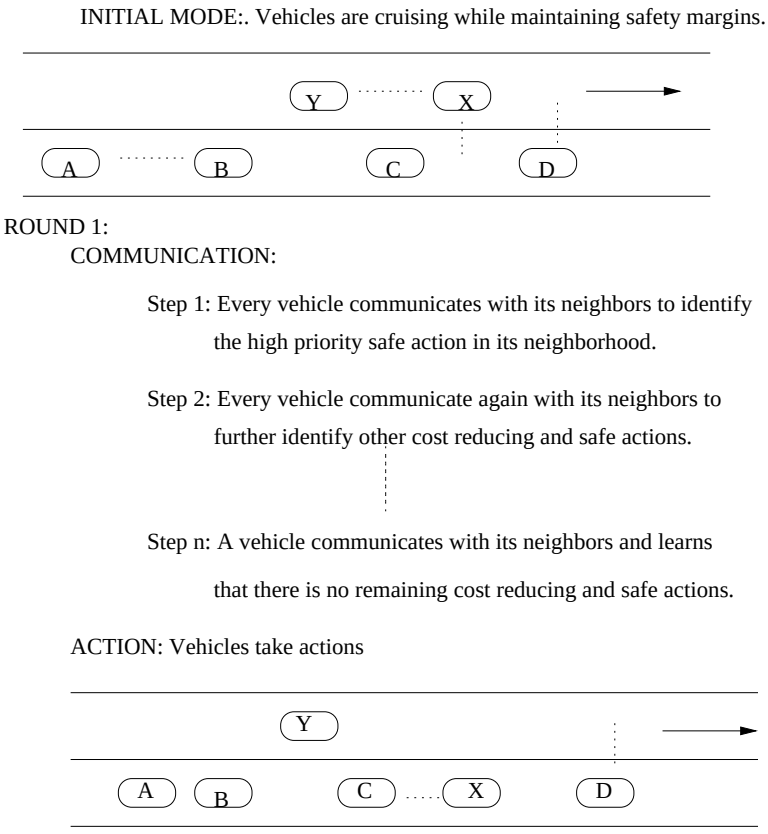

Figure 2.14: *This schematic depicts the rounds of* communication *and* action*. During each* round*, vehicles* communicate *at the coordination layer prior to taking* actions*. The coordination controller (algorithm) is implemented through different* steps *of* communication*. These steps of* communication *continue until the controller of each vehicle has determined a safe action for the vehicle to take during the* action *mode. In the above example, vehicle X wants to move to the right lane. Vehicle D wants to move to the left lane. Vehicles A and Y want to join their front vehicles. The vehicles* communicate *in one or more* steps *to resolve their safe coordinated* actions *at each* round*.*

2.3.2 Control Problem

We use the cost of a vehicle state to both identify the actions which need to be taken and prioritize those actions. Suppose the set of vehicles in a road link is denoted by the set $I = \{1, 2, \ldots, i, \ldots, n\}$ where each integer i corresponds to a single vehicle. The global minimization problem is the following. Denote by $\overline{u} = (u^1, \ldots, u^n)$ a vector of actions u^i for vehicles $i = 1, \ldots, n$ such that at least one vehicle i $u^i \neq cruise$. We only consider those $\overline{u}$ such that each u^i is *safe*, i.e., we require that $u^i \in \hat{U}^i$, $\forall i$ and denote those $\overline{u}$ by the set $\mathcal{U}$. Recall $C^i_{\overline{u}}$ denotes the cost of the new state of vehicle i after the actions specified by $\overline{u}$ have been completed. One possible objective of the coordination controller would be to minimize the global maximum cost, $max_{i \in I} C^i$. Let $\overline{u}_{min} \in \mathcal{U}$ be such that

$$max_{i \in I} C^i_{\overline{u}_{min}} = min_{\overline{u} \in \mathcal{U}} \{max_{i \in I} C^i_{\overline{u}}\}. \tag{2.3}$$

This is a global minimization problem. The minimizing action $\overline{u}_{min}$ may be the trivial action, i.e., there may be no safe non-trivial action $\overline{u}$ which decreases the cost of vehicles. Also, there may be more than one $\overline{u}_{min}$ such that (2.3) holds.

Minimizing the global cost above requires that each vehicle obtain information about the global configuration of vehicles on the link as well as their states and costs. They would then be able to choose a safe vector of actions $(u^1, \ldots, u^n)$ that minimizes the global cost. However, our communication scheme for the coordination controller of a vehicle only provides information about the neighbors of that vehicle. The following example illustrates that global information is necessary for global cost minimization.

Example 3 *Figure 2.15 depicts the case where the maximum platoon size is two. The global cost is defined as the total* space-time *taken by vehicles; vehicles* join *each other to*

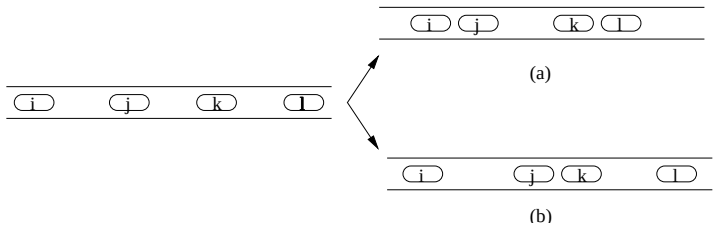

Figure 2.15: *If knowledge of global configuration is not provided to the vehicles, then cost minimization cannot be guaranteed.*

minimize the global cost. Assume that all vehicles are non-Exiting and the cost of a Single vehicle is higher than a Leader *or* Tail*. Thus, the global cost in the resulting case (a) is less than that of case (b).*

If vehicle j is only aware of its neighbors i and k through local communication, then it might randomly choose to join *either one and if it joins k, then l remains* Single *and global* space-time *utilized by the set of vehicles is not optimized as in case (b). If j could communicate with all vehicles and was aware of l, it would join i so that l could join k. Thus, with local communication both cases (a) and (b) are possible. However, if vehicles were aware of their global topology through global communication, and moreover, if they could globally negotiate their possible actions and the resulting global topology together, then they could choose appropriate* join *actions to achieve (a).*

We have just seen that the global minimization scheme described above cannot be implemented, given the limitations of our communication scheme. We now define a local minimization problem that allows a vehicle to identify and select an action such that the overall action $\overline{u} = (u^1, \ldots, u^n)$ is safe and reduces cost.

The problem at hand, therefore, is to devise -within our constraints of having

only *local* information- a systematic method to create a complete vector of actions $\overline{u} = (u^1, \ldots, u^n)$ which reduces cost. Moreover, we must at all times guarantee the *safety* of the vehicle actions $\overline{u} = \{u^1, \ldots, u^n\}$. We now formulate a local minimization problem for each vehicle. Recall that $C^j_{\overline{u}_i}$ denotes the cost of vehicle j after the coordinated action $\overline{u}_i$. The problem for an individual vehicle i is to find a safe coordinated action $\overline{u}^*_i$ such that the maximum cost in its neighborhood is minimized,

$$max_{j \in \mathcal{N}^i} C^j_{\overline{u}^*_i} = min_{\overline{u}_i}(\ max_{j \in \mathcal{N}^i} C^j_{\overline{u}_i}). \tag{2.4}$$

Note that in comparison with (2.3) the maximum in (2.4) is taken only over the neighborhood of i.

We note that it is possible for two or more vehicles in a single neighborhood to be involved in distinct coordinated actions as shown in Example 4 below.

Example 4 *In Figure 2.7 it is possible that vehicle A joins B as vehicle X is moving to the right lane. The cost of vehicle B is only affected by the coordinated action of A and B* join*ing as long as the actions of all vehicles are safe.*

However, our algorithm requires that the same vehicle cannot be involved in more than one coordinated action. We describe below our algorithm which incorporates this local minimization problem.

2.3.3 Local Cost Reducing Algorithm

We now propose an algorithm that allows vehicles to solve their individual coordination control problems in a way that collectively and systematically contributes to solving (2.4).

We use the cost values to prioritize the vehicle actions as in Section 2.2.4. A higher cost reflects a higher priority. When there are two or more vehicles with the same cost, we use an arbitrary discriminating parameter to order the priority. Specifically, we assume as above that the road link labels vehicles from 1 to n and the value of the integer label is used to resolve the tie, i.e., the vehicle with the larger label has priority.

At each step of communication, the vehicles communicate with each other and exchange information regarding their current states, costs, and safety constraints. Knowing the states and costs of neighbors allows vehicle i to learn whether a coordinated action increases or decreases the costs of its neighbors and then to choose if possible an appropriate action that does not increase the cost of its neighbors. Specifically, every neighbor considers the vector of safe actions that it could take, and the co-maneuvering actions that it would require from its neighbors. It then considers the impact of such actions on the cost of itself and its neighbors. Consequently, every vehicle i *plans* a coordinated action $\overline{u}_i^*$ that minimizes its cost,

$$C^i_{\overline{u}_i^*} = min_{\overline{u}_i}\{C^i_{\overline{u}_i}\} \tag{2.5}$$

and does not increase the cost of neighbors of i. That is, if $C^j < C^i$, then $C^j_{\overline{u}_i^*} < C^i_{\overline{u}_i^*}$, and if $C^j > C^i$, then $C^j_{\overline{u}_i^*} \leq C^j$, $\forall j \in \mathcal{N}^i$. If there is no action that could reduce the cost as above, then we say that vehicle i has no *planned* action.

We want the vehicles to assume priority for the actions that decrease the highest costs in their neighborhoods.

Definition 2 *A vehicle j, $j \in \mathcal{N}^i$, has the* local maximum cost *in the neighborhood of i if* $C^j = max_{k\in\mathcal{N}^i} C^k$.

We want every vehicle to identify the neighbor with the local maximum cost in its neighborhood and coordinate its action with that neighbor in order to minimize the local maximum cost in its neighborhood. A vehicle must only take safe actions. As such, we require that every vehicle i follows these steps where the implementation procedure for these steps will be explained in details later:

- I. Calculate the set of safe actions and, if possible, *plan* a coordinated action $\overline{u}_i^*$ that minimizes the cost as described by (2.5).
- II. Communicate with other vehicles and find a neighbor, say, j, with a planned coordinated action, $\overline{u}_j^*$, which has the highest cost priority described by (2.4).
- III. Verify the safety of $\overline{u}_j^*$ described by (2.1) through communication.
- IV. If $i \in \mathcal{N}_c^j(\overline{u}_j^*)$, then the action of i is determined accordingly to be coordinating with u^j.
- V. If no action is determined by Step IV because of higher priority of other vehicles or because of a lack of safety, repeat the above Steps I to IV to determine a safe cost reducing action. If there is no safe cost reducing action to take, then take $u^i = cruise$.

We define an auxiliary parameter that allows vehicles to implement II.

Definition 3 *The* priority permission *of i is a parameter with values in $\mathcal{N}^i$ whose value is defined to be j iff*

- *j is a neighbor of i, $j \in \mathcal{N}^i$, and*

- *j has the maximum cost amongst the neighbors of i who have* planned coordinated actions*. If there are neighbors with* planned coordinated actions *and equal maximum costs, then j has the largest label among them.*

If there is no j to meet the above conditions, then the priority permission of i is not defined.

We denote the priority permission of i with ψ^i. The values of priority permission parameters are exchanged among neighbors through communication. When $\psi^i = j$ we say that i sends a priority permission to j and informs its other neighbors about its priority permission value at each communication step shown in Figure 2.14. If ψ^i is not defined, then i would send $\psi^i =$ "Nil" to its neighbors.

Definition 4 *The coordinated action of i, $\overline{u}_i$, has* local priority *iff $\psi^j = i, \forall j \in \mathcal{N}^i$.*

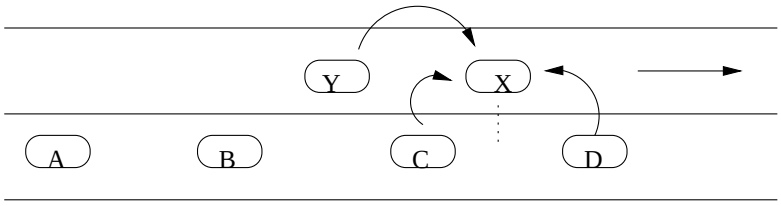

Figure 2.16: *Every neighbor of X sends a priority permission to it where a priority permission is shown by an arrow.*

We will show later how vehicles determine that it is safe for different coordinated actions, all with local priority, to take place simultaneously. We assert the following lemma.

Lemma 1 *A vehicle can be involved in at most one coordinated action at a time.*

Proof: The proof is by contradiction. Let $\overline{u}_i$ and $\overline{u}_j$ both have local priority where $i \neq j$. Suppose that k is involved in both coordinated actions, $k \in \mathcal{N}_c^i(u^i) \cap \mathcal{N}_c^j(u^j)$. Then,

$k \in \mathcal{N}^i \cap \mathcal{N}^j$. Since $\overline{u}_i$ has local priority, by definition of local priority we must have $\psi^k = i$, and similarly, we must have $\psi^k = j$. This is a contradiction because ψ^k has a unique value.

$\natural$

A question that one may ask is whether this scheme of giving and receiving priority permissions could cause a *livelock*. A livelock is a situation in which a process is not explicitly blocked, but some critical stage of it is unable to finish. Figure 2.17 illustrates a possible livelock situation where every vehicle sends its priority permission to a neighbor, but no action has local priority because no vehicle receives permissions from all its neighbors. An

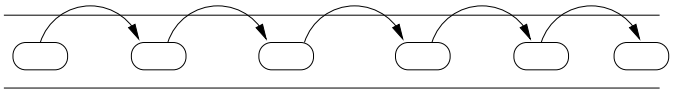

Figure 2.17: *A livelock may happen if every vehicle passes its priority permission to a neighbor but not to the same vehicle. No action is taken because no vehicle receives the priority permission of all its neighbors.*

action that does not have local priority will not be commanded to the regulation layer. We show in the following lemma that livelock is not a problem in our algorithm when there are a finite number of vehicle states and cost values.

Lemma 2 *Assume there exists a vehicle i with a planned coordinated action $\overline{u}_i$. Then there is at least one vehicle whose coordinated action has* local priority*.*

Proof: Let Q be the set of vehicles who plan coordinated actions, $Q \subseteq I$. By hypothesis, $Q \neq \emptyset$. Consider the set $Q_{max} \subseteq Q$ consisting of the vehicles with the maximum cost in the set Q. There must be such a maximum since Q is finite. Take the vehicle in Q_{max} of highest ordering by the integer labeling. Call it k. We claim that this vehicle k must

receive *priority permissions* from all its neighbors. Let $i \in \mathcal{N}^k$. We want to show $\psi^i = k$. By construction, $C^k \geq C^j$, for all $j \in \mathcal{N}^i \cap Q$. So, if $C^j < C^k$, $\forall j \in \mathcal{N}^i \cap Q$, then $\psi^i = k$. If there exists a $j \in \mathcal{N}^i \cap Q_{max}$, such that $C^j = C^k$, then by definition of *priority permission*, $\psi^i = k$ because $k > j$, since we picked k to have the highest integer label in I. Thus, $\forall i \in \mathcal{N}^k$, $\psi^i = k$, and $\overline{u}_k$ by definition has local priority. $\sharp$

We also need to ensure that simultaneous coordinated actions are taken safely. This is because a vehicle plans a coordinated action in its neighborhood without any knowledge of the other neighborhoods. Coordination safety must be verified when different vehicles receive *local priority* for their coordinated actions simultaneously. In Figure 2.18 vehicle C coordinates its action with X and vehicle B coordinates its action with A. However, B and C need to coordinate their actions together also, since B and C are neighbors. In the following, we discuss how vehicles verify the safety of their *planned* coordinated actions through communication and implement III. Suppose now we have determined, by the algorithm described thus far, a finite number of vehicles $\{i_1, \ldots, i_l\} = \mathcal{I}$ whose planned coordinated actions have local priorities. Let $\overline{u} = (u^1, \ldots, u^n)$ be defined as follows. For a vehicle $i_k \in \mathcal{I}$, the action u^{i_k} is defined by the planned coordinated action $\overline{u}_{i_k}$. For a vehicle j, if $j \in \mathcal{N}_c^{i_k}(u^{i_k})$, some $i_k \in \mathcal{I}$, then u^j is the co-maneuvering action as determined by the action u^{i_k}. (Recall from Lemma 1 that any vehicle can only be involved at most one such coordinated action.) If $j \notin \mathcal{N}_c^{i_k}(u^{i_k})$, $\forall k, 1 \leq k \leq l$, then we define $u^j = cruise$. We would like to verify whether $\overline{u}$ is safe. Again, we only have local communication at our disposal in order to verify safety. Thus, we require vehicles involved in every coordinated action to verify the safety of their *planned* actions through local communication. That is,

when vehicle i plans a coordinated action $\overline{u}_i$, it verifies through (a) communicating with its neighbors, and (b) its co-maneuvering neighbors communicating with their neighbors, that $\overline{u}_i$ is safe.

Definition 5 *Let $\{i_1, \ldots, i_l\} = \mathcal{I}$ and the vector of actions for all vehicles $\overline{u}$ be as above. Let $\{\overline{u}_{i_k}\}_{1 \leq k \leq l}$ be the set of planned coordinated actions of vehicles in $\mathcal{I}$. We say that the planned coordinated action of i_k, $\overline{u}_{i_k}$, is* safe *iff u^{i_k} is safe, i.e., $u^{i_k} \in \hat{U}_{i_k}$, and every co-maneuvering action u^j is safe, i.e., $u^j \in \hat{U}^j$, $\forall j \in \mathcal{N}_c^{i_k}(u^{i_k})$ with respect to $\overline{u}$.*

We define another auxiliary parameter called a *safety permission* which we use to ensure that different planned coordinated actions are indeed safe to be taken simultaneously and would not interfere with one another. Specifically, the safety of a planned coordinated action is verified through *safety permissions*.

Definition 6 *Let j be a co-maneuvering neighbor of i. The vehicle j provides a* safety permission *to i iff the co-maneuvering action u^j is safe, i.e., $u^j \in \hat{U}^j$ with respect to the $\overline{u}$ defined above.*

Definition 7 *The coordinated action $\overline{u}_i$ is* confirmed *if $u^i \in \hat{U}^i$ and i obtains safety permissions from every co-maneuvering neighbor j, $\forall j \in \mathcal{N}_c^i(u^i)$.*

Suppose j is a co-maneuvering neighbor of i and u^j is not safe with respect to the planned actions of its neighbors, say, $k_1, \ldots, k_l$. Let C_j denote the maximum cost being reduced by the coordinated action that involves i as described in 2.4. Define $C_{k_1}, \ldots, C_{k_l}$ similarly. Then j and its neighbors compare the costs they are reducing, C_j and $C_{k_1}, \ldots, C_{k_l}$. If C_j is the largest cost being reduced, then j "wins" and provides safety permission to i

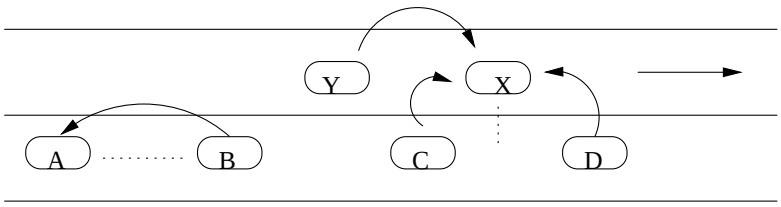

Figure 2.18: *Neighbors of X send priority permissions to it while A receives priority permission of its neighbors, too. To ensure that the co-maneuvering actions of B and C do not interfere, we require safety permissions.*

while $k_1, \ldots, k_l$ provide "Nil" to the vehicles of which they are co-maneuvering neighbors. In case of equal costs, i.e., if $C_j = C_{k_s}$, $1 \leq s \leq l$, then we use a tie resolution arbitration. That is, we allow vehicles to compare the integer values of the vehicles whose costs are being reduced where the highest integer "wins".

This scheme of safety verification will not result in a livelock. We prove this below.

Lemma 3 *Suppose that there exists a vehicle i whose coordinated action has local priority. Then at least one vehicle will have a* confirmed *coordinated action.*

Proof: The proof is similar to that of Lemma 2. Let Q denote the set of vehicles whose coordinated actions have local priority. By hypothesis, $Q \neq \emptyset$. Recall that an action is *confirmed* when the vehicle receives *safety permissions* from all its co-maneuvering neighbors. Again, take the set $Q \subseteq I$ of vehicles with local priority , $Q = \{j : \psi^l = j, \forall l \in \mathcal{N}^j\}$. Consider the set $Q_{max} \subseteq Q$ consisting of the vehicles with the maximum cost in the set Q. Then take the vehicle of the highest ordering by the integer labeling. Call it k. We claim that this vehicle k must also receive *safety permissions* from all its co-maneuvering neighbors and, thus, has a *confirmed* action. Let $i \in \mathcal{N}_c^k(u^k)$. We want to show i sends a

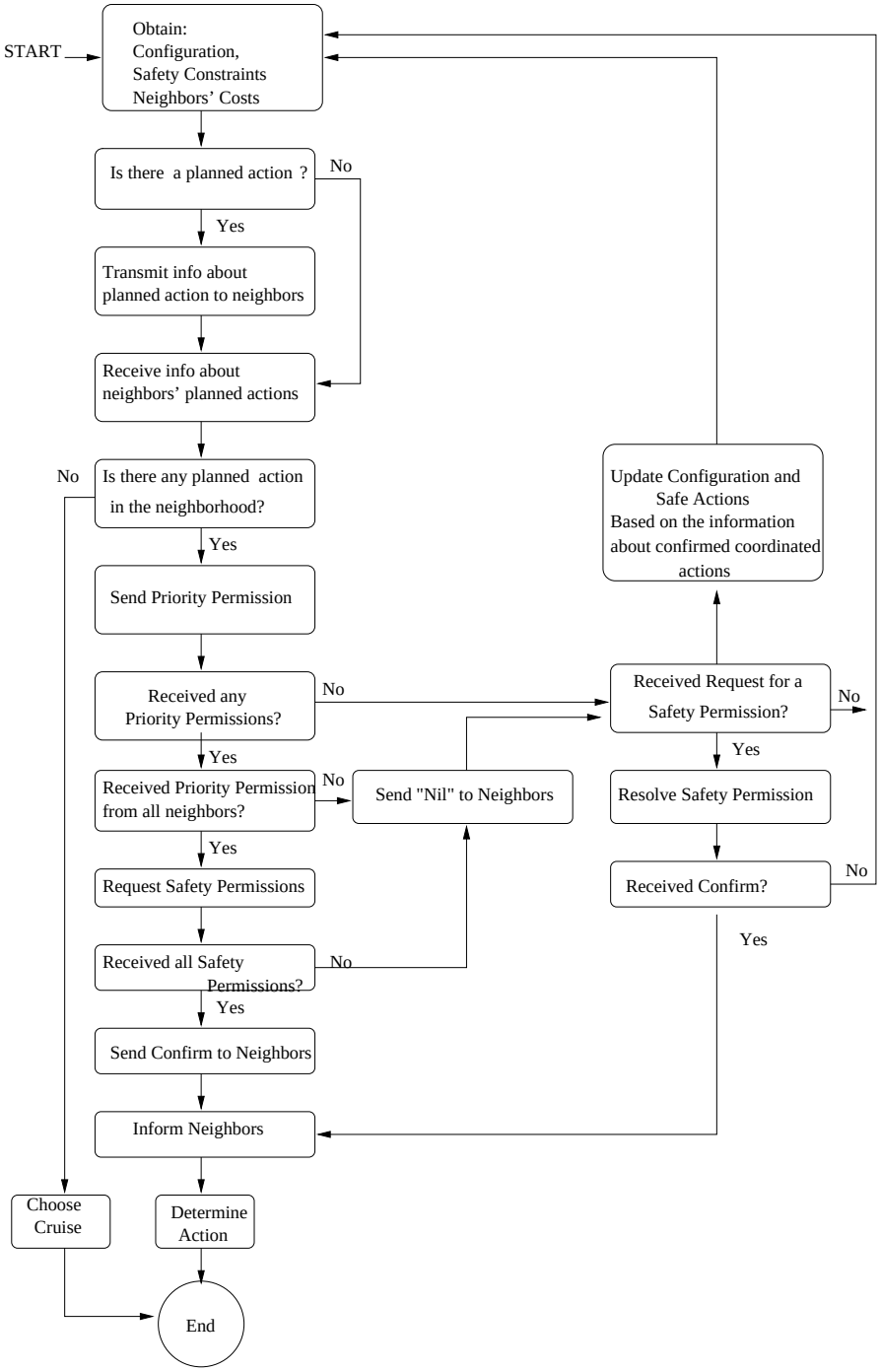

Figure 2.19: *Algorithm flowchart for local cost minimization algorithm. The algorithm procedure from START to a point of return to START is called one* step *and every vehicle will go through at least one* step.

safety permission to k. By construction, $C^k \geq C^l$, for all $l \in Q$ with approved actions. If there exists a neighbor of i, say j, where the actions u^i and u^j are not safe with respect to one another, then either the maximum cost in the neighborhood of i is greater than the the maximum cost in the neighborhood of j, or they both have the same maximum cost in their neighborhoods but $k > l$, $\forall l \in Q$. Therefore, by definition of *safety permission* u^i will have priority and i will send *safety permission* to k. ♯

Thus, the vehicles will verify the priority and safety of their planned coordinated actions through priority and safety permissions, respectively. This outlines the procedure of one step of the communication in one *round*. The flowchart for the local cost reducing algorithm is depicted in Figure 2.19. The algorithm procedure from the "Start" to the "End" or from the "Start" to a point that returns to the "Start" is considered a *step*. At the end of the first *step* some coordinated actions are *confirmed* and, thus, the actions of some vehicles are determined. At the next step, the same procedure is repeated where the actions that have already been determined by the previous steps add to the safety constraints of future planned actions. The *steps* continue until no further cost reducing and safe actions can be taken. At that point, those vehicles whose actions are not yet determined, will choose to *cruise*.

Lemma 4 *Every vehicle communicates a finite number of messages at each* step *of the communication procedure.*

Proof: At each step of the communication procedure, a vehicle exchanges messages with its neighbors regarding its cost, position, and planned actions, as well as priority and safety permissions. The number of messages that a vehicle transmits to and/or receives from a

neighbor is finite. The number of neighbors for a vehicle is also finite. Thus, the total number of exchanged messages for a vehicle is finite. ♯

Note that the communication messages that implement the algorithm include both positive and negative acknowledgments in order to eliminate the possibility of a deadlock. Deadlock is a situation in which two or more processes are unable to proceed because each is waiting for one of the others to proceed. After a request for *priority (safety) permissions*, we require vehicles to send either the related permission, or the message "Nil" to the request sender. Similarly, when a vehicle receives both *priority* and *safety permissions* but does not have a *confirmed* action, it sends a "Nil" message to its neighbors to avoid deadlocks. Finally, when a vehicle learns that its coordinated action is safe, it sends a *confirm* message to the co-maneuvering neighbors and informs other neighbors. However, if the vehicle does not receive the required safety permissions, it send a "Nil" to the neighbors.

Lemma 5 *The algorithm for each vehicle takes a finite number of steps to determine an action.*

Proof: If there is no safe, cost reducing action for any of the vehicles, the algorithm ends for all vehicles after the first step. Suppose there exist some vehicles with coordinated actions. The proofs of Lemmas 2 and 3 show that at every step of the algorithm -assuming there is safe, cost-reducing action for at least one vehicle- there is a vehicle whose action is determined and *confirmed*. Since the number of vehicles is finite, it takes a finite number of algorithm step to determine the action of all vehicles. ♯

Lemma 6 *The local cost reducing algorithm procedure is instantaneous and takes a negligible amount of time.*

Proof: The proofs of Lemmas 4 and 5 show that the number of communication messages for every vehicle during the algorithm procedure, i.e., prior to taking an action, is finite. By Assumption 2, the time for a finite number of communication messages is negligible. Thus, the algorithm procedure is nearly instantaneous. ♯

By definition of planned actions, the actions determined by this algorithm minimize the local maximum cost values. We now see that when possible, the algorithm reduces the global maximum cost.

Lemma 7 *If the global maximum cost $max_{i\in I}C^i$ can be reduced safely by a coordinated action $\overline{u}_k$, then the local cost reducing algorithm will reduce $max_{i\in I}C^i$.*

Proof: Let Q be the set of vehicles which have the global maximum cost and have a planned coordinated action. That is, if $j \in Q$, $C^j = max_{i\in I}C^i$ and there exists a safe planned coordinated action $\overline{u}_j$. In particular, we assume that there exists at least one vehicle with a planned coordinated action. By Lemmas 3 and 4, there exists at least one vehicle (not necessarily with the global maximum cost) with a confirmed action. Let P be the set of vehicles with confirmed actions. By the above, $P \neq \emptyset$. We claim now that at least one vehicle in P has global maximum cost, i.e., $P \cap Q \neq \emptyset$. Let $Q_{max} \subset Q$ denote the set of vehicles in Q that have the largest cost. Let k be the vehicle in Q_{max} with the largest label. The proofs of Lemmas 2 and 3 show k will receive *priority* and *safety permissions* from all its neighbors and, thus, has a confirmed action. ♯

Finally, the above lemmas and their proofs result in the following theorem.

Theorem 1 *The local cost reducing algorithm instantaneously results in vehicle actions that (a) are safe and reduce vehicle costs and (b) reduces the maximum global cost when*

such reduction is possible through safe coordinated actions.

The cost reducing algorithm reduces the cost of vehicles safely, but also it is practically feasible because it requires local communication. In our local communication structure every vehicle communicates with its neighbors. We have assumed that every vehicle learns and updates its configuration and can communicate with its neighbors. However, the vehicle's awareness of its neighbors and their relative positions, i.e., its configuration, is not trivial and needs to be discussed. Furthermore, learning the communication addresses of the neighbors is equally important and needs close attention. These communication issues are discussed in the next chapters.

Chapter 3

Controller Implementation: Communication

Communication is essential to implement the coordination controller. An automated vehicle communicates with its neighboring vehicles to learn its current configuration and run the coordination controller algorithm to find a safe action as discussed in the previous chapter. In this chapter, we discuss communication at the coordination layer. First, we summarize the communication structure in AHS. Then we turn our attention to the issue of address resolution and review the existing work. Finally, we provide innovative solutions for address resolution in one-lane and multi-lane automated highway systems.

3.1 Communication Structure Hierarchy

We consider a hierarchy in the communication structure as depicted in Figure 3.2. The hierarchy divides the communication structure into two sets of networks. A

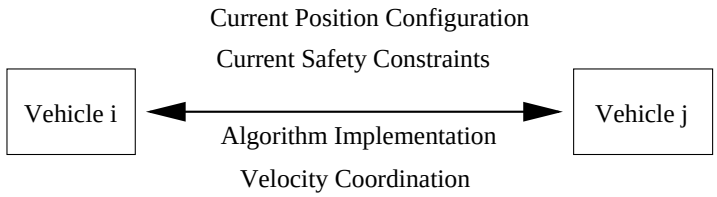

Figure 3.1: *Communication plays a crucial role in coordination and safety of automated vehicles.*

local area network (LAN) is considered for communication amongst vehicles in a platoon. A wide area network (WAN) accommodates communication among the platoon *Leaders* and *Single* vehicles [10, 19]. The communication hierarchy reduces the amount of required

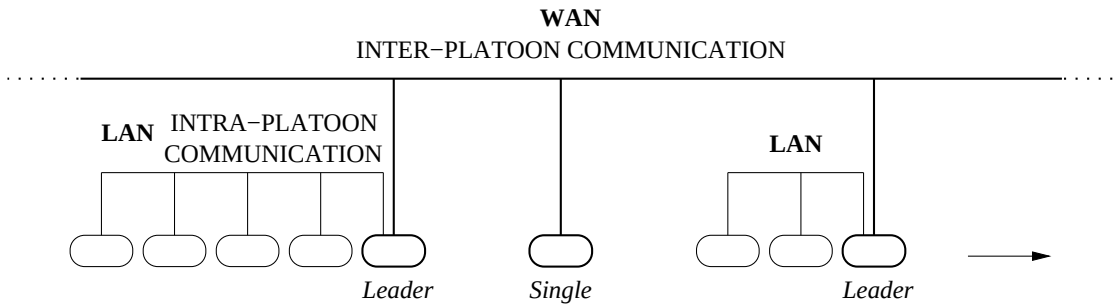

Figure 3.2: *Communication hierarchy*

power and bandwidth by decreasing the number of direct vehicle-to-vehicle communications. In this hierarchical structure, a *Leader* needs to communicate with its LAN and other WAN members only. Therefore, it need not spend power and bandwidth on resolving communication addresses of every neighboring vehicle in other LANs. Similarly, its *Follower* vehicles use power and bandwidth only on communicating within the LAN.

The communication medium for LAN could be infra-red or radio signals. Different LAN structure designs are studied in [20, 19, 21]. The LAN is mainly used by the regu-

lation controllers. The communication medium for WAN is radio signals and we use it to implement the coordination controllers. In the rest of this chapter we discuss the issue of address resolution in the WAN.

3.2 Address Resolution

In any communication network, a user has a logical identifier (its IP address or social security number) and a physical location. The mapping between the logical identifier of a user and its physical location is called *binding*. An *address resolution protocol* (ARP) is a distributed algorithm that defines the binding. In AHS applications, the physical address is determined by relative position, e.g., the vehicle that is immediately in front of my vehicle, and the logical address is determined by an absolute address, e.g., the vehicle's license plate number. Address resolution for the coordination layer is challenging because the mobile network is continuously changing its topology.

The address resolution protocol in conventional networks assumes that a computer is aware of its own binding, i.e., it knows its own network address and absolute address or identity. Hence, if one can address a computer by its absolute address, then one can ask for its network address and vice versa.

In the AHS context, each vehicle knows its absolute address, e.g., license plate number. Suppose now that vehicle A wishes to learn the network address of the vehicle in front of it. If it knows the latter's absolute address or identity (say, it is B) it can broadcast a message, "What is vehicle B's network address?" and then B could reply with its network address. Unfortunately, if all that A knows is that it wishes to send a message to the "vehicle

in front of A", it cannot do so. Note that "vehicle in front of A" uniquely identifies B, but neither A nor B know that B is that vehicle. Indeed, if A broadcasts the message "who is in front of A?", all vehicles in A's neighborhood will hear this message, but will not be able to figure out which one of them is in front of A. This is because the binding information that is important for every vehicle in the coordination layer is the mapping between the relative position of a vehicle and its logical identifier, i.e., $< relative\ position, ID >$. This research has benefited from the work that has already been done on the coordination layer [1, 10, 9, 8]. However, the work on the coordination layer assumes that communication address is not a problem and builds on the assumption that address resolution protocols are available.

3.3 Existing Work

When address resolution is initially provided but the binding between the logical identifier and the physical location of a user is not fixed and changes over time then the problem of *location management* arises which is one of the most fundamental problems facing mobile networking today: How does the network know where the intended recipient of a message is currently located? Who should be responsible for determining the user's location? The issue of binding is being discussed in the context of the Internet, cellular telephony, and ad-hoc networks [22].

3.3.1 Mobile IP

In the Internet protocol (IP) there is a fixed relationship between a computer's IP address and its physical location, i.e., the binding is fixed over time [23]. Moreover, IP uses a hierarchical scheme to support scalability. The hierarchy that is seen in the logical identifier of a user in general reflects the physical connection of that user to the network. For example, the IP address of a host is divided into two levels of hierarchy: a network number identifying the network to which the host is connected, and a host number identifying the particular host within that network. Routers within the Internet route packets based on the network number of the destination address in each packet. Once the packet reaches that network, it is then delivered to the correct individual host on that network.

Source routing, the insertion of routing information into a datagram by the node from which it originates, is a proposal to overcome the problem of hierarchical routing based on physical location. A source that is aware of the mobility of another user will specify the mobile's current physical address each time it sends a message and will not rely on its logical address, hence the term source routing. Strict source routing specifies the route hop-by-hop while loose source routing specifies the end user [22].

Considerable work has been done concerning mobile IP and Figure 3.3 shows one routing scenario for mobile IP users. However, there is not a standard set as of the time of this writing. One of the fundamental problems that faces the designers is compatibility of the proposed standard with the existing infrastructure. It is important for a mobile IP proposal to support addressing and routing to mobile hosts from existing correspondent hosts that have not been modified to support mobility and seem likely to remain so for

some time [22].

3.3.2 Cellular Telephony

The issues facing cellular telephony are similar to the problems of mobile IP and they in fact both use similar solutions to overcome the location management problem. For example, in most mobile addressing scenarios a mobile user has a home agent that maintains current location information for the mobile user and receives packets for it and *tunnels* them to the mobile user at its current location. We can draw an analogy between cellular telephony and mobile IP. A mobile terminal has permanent association with a home location register (HLR) and uses the service of a visitor location register (VLR) as it moves. HLR and VLR are analogous to home agent and foreign agent for a mobile terminal. There is also *call forwarding* by HLR that is similar to tunneling since it is HLR that has to find the current location of the mobile terminal [24].

There are yet differences between the two technologies in terms of the acceptable delay in communication which in turn determines the acceptable delay in address resolution. The acceptable amount of communication delay for each technology depends on the type of communication service that it provides. For example, telephony is more sensitive to delay. Therefore, real-time ubiquitous binding and bandwidth are basic issues for cellular telephony with its increasing number of users. However, the buffers in communication networks can store packets until a valid binding is achieved. For similar reasons the bandwidth is assumed to be able to accommodate all potential users and not to be an issue in the near future.

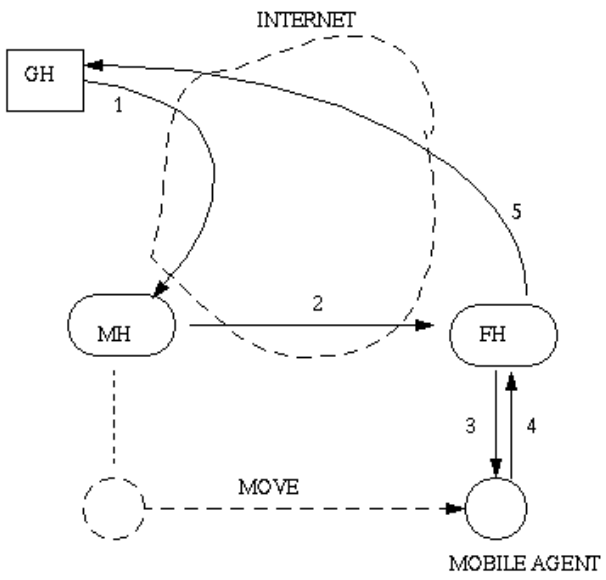

Figure 3.3: *A general host (GH) sends a message for the mobile agent to the mobile home (MH). The mobile home tunnels the message to the mobile agent via its foreign host (FH).*

Hierarchical Topology and Support Stations

Mobile IP and cellular telephony have a couple of major similarities. (I) They both use a hierarchical topology for network and addressing. (II) They both assume existence of at least some stationary stations such as a home agent, foreign agent, and routers.

The above assumptions are not true for the type of mobile networks that we are interested in. We are interested in a self-starting and self-organizing network of automated vehicles that is independent of the number of users and does not rely on road infrastructure. In a self-starting mobile network of automated vehicles, there is no hierarchical topology among the mobile users and the network is flat. More importantly, in a self-organizing mobile network there is no support station of any kind that its services of routing or redirecting could be assumed. The network of our interest is a mobile ad-hoc network.

3.3.3 Ad-hoc Networks

An *ad-hoc network* is the cooperative engagement of a collection of mobile hosts without the required intervention of any centralized access point [22]. It is an example of a flat network with no mobile support station.

From a graph theoretical point of view, an ad-hoc network is a graph, $G(N, E_{(t)})$, formed by denoting each mobile host by a node and drawing an edge between two nodes if they are in direct communication range of each other. The set of edges, $E_{(t)}$, so formed, is a function of time, and it keeps changing as nodes in the ad-hoc network move around.

Current routing protocols do not accommodate the self-starting behavior needed for ad-hoc networks. There have been routing methods proposed for ad-hoc networks that take an approach similar to the shortest path computation method [22]. Each node maintains a view of the network topology with a cost for each link. To keep these views consistent, each node periodically broadcasts the costs of its outgoing links to all other nodes using a multi-cast protocol such as flooding. As a node receives this information, it updates its view of the network topology and applies a shortest path algorithm to choose its next hop for each destination. Each node has a routing table that lists all available destinations and the number of hops to each.

Each route table entry is tagged with a sequence number which is originated by the destination station. To maintain the consistency of routing tables in a dynamically varying topology, each station periodically transmits updates, as well as immediately transmitting updates when significant new information is available. However, there is no notion of synchronization and no assumption about the update periods between the mobile hosts. These

packets indicate which stations are accessible from each station and the number of hops necessary to reach these accessible stations. The packets may be transmitted containing either MAC layer or network layer addresses. However, some of the link costs in a node's view can be incorrect because of long propagation delays, the partitioned network, etc. that might lead to formation of routing loops.

3.3.4 Location Information

The distinguishing characteristic of address resolution in automated highways is the importance of *location* in binding. All other mobile schemes start with the *logical identifier* and address resolution is to find the binding map to the location. In an automated highway, the address resolution protocols shall start with the (relative) location of a vehicle and map it to a logical identifier to provide binding. Hence, tunneling and message forwarding are of no use in this scheme.

3.4 Address Resolution in One-Lane AHS

Recall that the binding information for every vehicle in the coordination layer is the mapping between the relative position of a vehicle and its logical identifier, i.e., $< relative\ position, ID >$. In a one-lane automated highway, the relative position means either immediately in front or immediately behind, called *front* and *behind*, respectively, for abbreviation. The ID could be the IP address or license plate number. Thus, a WAN user as shown in Figure 3.2 needs to know $< front, ID >$ and $< behind, ID >$ since *front* and *behind* are the only neighbors that matter to it from the coordination point of view.

3.4.1 Basic Requirement

Puri et al [25] discuss what information is needed to build an address resolution protocol for vehicles in automated highways and show that inter-vehicle distances are insufficient and absolute position information are required. Assume that each vehicle has the relative x- and y- coordinates of vehicles in a circle around it, and the communication is local and is not biased in any direction. Then Puri et al [25] show that the problem of finding the vehicle in front becomes unsolvable under these conditions.

Puri et al show that relative coordinates with local communication are insufficient for address resolution and suggests the use of absolute position. However, we will show that in a one-lane automated highway, absolute position is not necessary to resolve the relative position. An external point of reference and communication are sufficient to provide the information about relative positions of vehicles. The external point of reference is provided through vehicle-to-road communication.

The vehicle-to-road communication is possible through *back-scattering* [26] communication which allows a fixed communication device on the road, called a *reader*, communicate with the vehicles that are moving at freeway speed. The term back-scattering refers to the fact that unmodulated sinewaves are generated by the reader and are modulated and back-scattered by the vehicle communication device.

3.4.2 Modeling

We introduce some notation relating to the WAN. The term *agent* refers to an automated vehicle that may be a *Single* vehicle or a platoon *Leader* because only *Single*

vehicles and platoon *Leaders* are active in the WAN.

We defined *configuration* and *neighbors* of a vehicle in the previous chapter. Let R_t^A denote the configuration of A in real time where every neighbor in this configuration model is represented by its communication address binding as the following.

$$R_t^A = (< self, ID >, < front, ID >, < behind, ID >).$$

The parameter "ID" contains the the communication address of the agent and can either take the value of an address or *"unknown"*. We add the binary parameter "exists" to represent the existence of a neighbor in the configuration.

$$R_t^A = (< self, ID >, < front, exists, ID >, < behind, exists, ID >).$$

Our address resolution scheme relies on protocol exchange among agents, and communication failure must be considered. While we present no solution to overcome such failure, we propose a solution for awareness of such cases. Before taking actions, agents should verify their models with one another, and an agent who is aware of a possible inaccurate model of the configuration by itself or a neighbor will not take any action.

During an action, the configuration update could take as long as the action takes to be carried out by the regulation layer. The solution to real-time configuration update is to reflect the action process in the model by a binary parameter that represents the action status.

$$R_t^A = (< self, ID, busy >, < front, exists, ID, busy >, < behind, exists, ID, busy >)$$

Accuracy of the configuration model is represented by the binary parameter "*flag*". Since the information about the *front* agent is independent of the information about the

behind agent, there are two independent *flag* parameters in the model.

$$R_i^A = (< self, ID >, < front, flag, exists, ID, busy >, < behind, flag, exists, ID, busy >) \tag{3.1}$$

The *flag* parameter, when set to "1", verifies the correctness of the "exists" and "ID" parameters.

A vehicle updates its configuration model only when it is *Leader* or *Single*, i.e., it has access to the WAN. A *Follower* could *split* from a platoon and become an agent, in which case its model is updated.

The model above represents the status of an agent A and its neighbors. We propose an *addresser* for every agent to be in charge of the configuration model as described above in 3.1. Addresser of A communicates with other agents on the road and updates the above modeling parameters. Note that while in practice controller and addresser could be combined, they have different roles. The controller is in charge of controlling the actions, while the addresser provides the network communications addresses to the controller. The controller's communication range is limited to the *front* and *behind* agents in the WAN for coordination purposes, while the addresser can contact many agents in a wide range of communication that is limited only by hardware capability and power and interference problems. Finally, the controller is in the coordination layer, while the addresser can have a range of services that go beyond the coordination layer. The controller provides the addresser with information about action status such as being requested, performing, done, etc., while the addresser provides the controller with the updated networking addresses.

3.4.3 Algorithm

We order agents, $\mathcal{A}$, in one lane $(\mathcal{A}, \leq)$, such that an agent A in front of agent B is "less" than B, $A \leq B$. We assign serial numbers to vehicles such that the order of serial numbers preserves the position order of vehicles. When two agents compare their serial numbers, the agent with a smaller serial number is in front of the other one. Consequently, a set of agents can compare their serial numbers and each would find the same exact order of agents. This provides a simple means of picturing their positions from front to back according to their ID numbers. For example, the agents in Figure 3.4 would be mapped: $A \longmapsto 1$, $B \longmapsto 2$, $C \longmapsto 3$. The integers correspond to the serial numbers of the agents. In practice, vehicles need to communicate their communication ID numbers along with their serial numbers. In the following discussions we make the simplifying assumption that serial numbers are identical to the communication ID numbers.

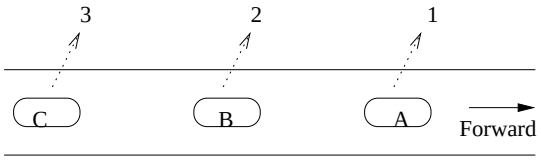

Figure 3.4: *Platoon* Leaders *are mapped to serial ID numbers where the order of numbers preserves their order of positions on the road.*

In section 3.2 we defined address resolution in the context of AHS as the information binding $< relative\ position, ID >$.

Theorem 2 *Address resolution of agents through local communication is possible by well-ordering them according to their position in one lane.*

Proof: The assumption is that we order the vehicles and provide ID numbers to them according to their position on the road such that every agent is assigned a number by the road which is greater than the number assigned to the vehicle in front of it and smaller than the number assigned to the vehicle behind it. Suppose an agent A wants to obtain the ID of its *front* neighbor. It transmits its ID and asks for ID numbers that correspond to the set of vehicles that are in front of it and receives responses. So it can obtain the ID of its *front* neighbor. Similarly, an agent can find its *behind* neighbor.

3.4.4 Implementation

The important principle of our address resolution scheme is breaking up the problem of address resolution into *initialization* and *updating*. Initialization is done externally by the road, i.e., an external point of reference, and it refers to assigning serial numbers to the agents. The vehicle-to-road communication that was discussed in 3.4.1 accommodates the assignment of serial numbers to the agents. Updating is an internal task of the network itself and is done through protocol communication.

Initialization Process

The assumption is that during initialization, no vehicle actions are allowed until vehicles find their place in the coordination layer as *Leaders*, receive their serial numbers, and become able to network and communicate. In addition to a reader, the road infrastructure includes the following. A vehicle-detector that detects the presence of a vehicle in real time, a speed-sensor that estimates the average speed of vehicles $\overline{v}$ in real time and a computer that receives their data. The computer measures the time lapse between two consecutive

vehicle detections t and, hence, calculates their approximate distance $d = \overline{v}.t$. The accuracy of $\overline{v}$ is not crucial in estimating the distance and realizing whether two consecutive vehicles belong to the same platoon or not. The large difference between inter- and intra-platoon longitudinal safety margins makes it possible to verify the *Leader* status of a vehicle. This information enables the road to assign serial numbers to the agents.

A vehicle that passes the initialization road station as a *Follower* is not active in the WAN, and hence, is not initialized. That is, it does not have a serial number and does not know the numbers of its neighbors. Once this vehicle *splits* from its platoon and becomes an agent itself, it needs a serial number that is unique and consistent with its position order and reflects its position in the line of agents on the road correctly . In addition, this vehicle that *splits* may have *Followers* who in turn may want to *split* later. The communication ID numbers to be assigned to these vehicles must reflect their relative positions on the road with no ambiguity. While we use integers for road initialization, we consider using real numbers for further initialization due to *split* actions.

An example would be useful in presenting the initialization process of a *Follower* by its platoon *Leaders*. Consider vehicle B in Figure 3.5 which is about to *split* from agent A. The agent A is to provide a proper serial number for B. Assume that the serial number assigned for agent A by road initialization is 5. Agent A would assign 5.9 to B if B is the first *Follower* to *split*. To the next *Follower* that *splits*, A would assign 5.8, and to next one 5.7 and so forth. This way, A can only assign 9 serial numbers, i.e., it can have a maximum of 9 *splits*. However, if we change the base for our numbering, we can increase it to any desired number.

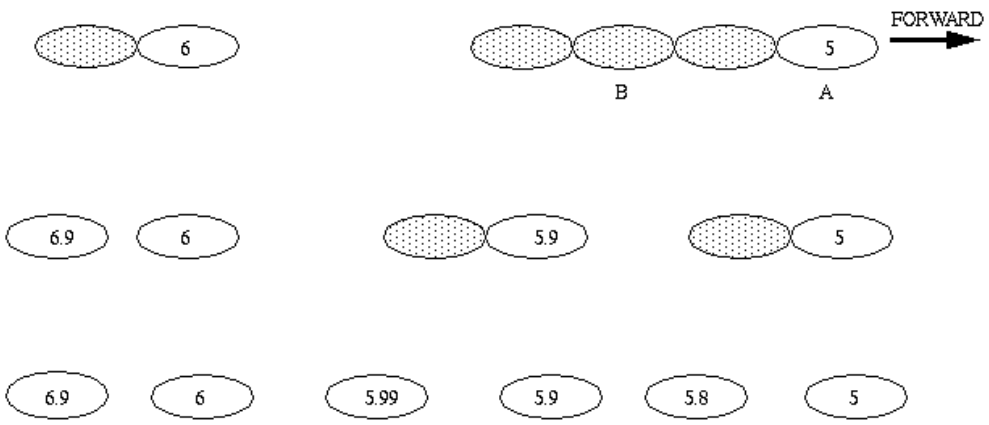

Figure 3.5: *Platoon* Leaders *initialize splitting vehicles.*

If a *Follower* is to *split* from B, it would be assigned 5.99 if it is the first one, 5.98 if it is the second one, and so forth. The agent number 5.99 would assign numbers from 5.999 to 5.991 to its splitting *Followers*. Note that the agent behind A would assign 6.9 to 6.1 to its splitting *Followers*, and so forth. This ID assignment algorithm keeps the serial numbers of the agents in accordance with their relative position on the road. Hence, comparison of serial numbers gives the relative position of agents with respect to one another.

Protocols

Assuming the initial network is established, this network itself can be utilized to update the addresses of agents as they take actions and change the topology of the network. Vehicles in a one-lane AHS change the network topology by *join* and *split* actions. However, the vehicle actions do not change the relative position of vehicles with respect to one another, but only change the grouping of vehicles into different platoons. Thus, there is no need to know the absolute position of vehicles, since the relative position is constant

for vehicles while it varies for agents. For example, if agent A is immediately in front of B and B is in front of C and so forth, it will always be so as long as there is no lane-change actions. Enter and exit can be considered as special cases of lane-change action and are not considered here.

Address updating is done by keeping track of the vehicle actions as they change their platoon groupings. The designed communication protocols relay the information about the actions and relative position of agents. Again, we assume no communication error or failure.

Assumption 3 *The communication channels are ideal and bidirectional.*

The process of address updating is implemented by the two types of *periodic* and *rendezvous* communication protocols. The periodic protocols could be of the following two types: *query* and *confirm*. After initialization, an agent, say A, actively searches for its neighbors as it travels along the road by sending a query message.

$$query \mid front \mid senderID$$

Disregarding possibilities of communication failure, if it comes within the communication range of an agent, that agent will respond by the following message.

$$query\ response \mid receiver\ ID \mid front \mid sender\ ID.$$

The agent A will recognize the responding agent as its *front* neighbor by comparing their serial numbers. Lack of any responses would mean lack of a neighbor. When there are multiple responses to A's query message, then A compares the serial ID numbers and finds out which one is immediately in front of it. Query is a multicast protocol.

A *confirm* protocol is exchanged between any *two* agents who are already aware of each other as neighbors and confirm their information about each other periodically. This allows vehicles to keep their configuration model updated and be aware of the changes (when confirm is not acknowledged). Each agent performs its own *confirm* protocols and responds to the *confirm* protocols of its neighbor. This is double checking of information by agents. The *confirm* message would contain the following information.

$$confirm \mid receiverID \mid front \mid sender\ ID,$$

and its response would contain:

$$confirmresponse \mid receiverID \mid front \mid sender\ ID.$$

Confirm is a unicast protocol which implies that the coordination layer has one-to-one channels available to the agents in addition to the multicast channel.

Rendezvous communication can be illustrated by an example as depicted in Figure 3.6. Assume agent B is behind A and requests to *join* A and the request is granted. Both A and B update their configuration models when the action is done. Meanwhile, A's and B's addressers cooperatively inform the addresser of the agent immediately behind B about the *join* action so that agent C realizes when its *front* agent is busy and also learns that after the action its *front* agent is no longer B, and updates it to A. In general, when an agent is engaged in an action, its neighbors are aware of its busy status and after the action they update their configuration parameters accordingly.

The proposed protocols work based on the assumption that initialization is provided to all agents as described in Section 3.4.4. The initialization mainly provides the

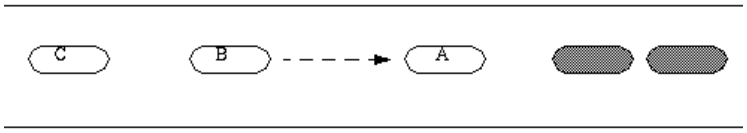

Figure 3.6: *Agent B wants to JOIN agent A, C is behind B, and its Front ID is affected by the JOIN action of B.*

basis for the protocols to work and that is a unique serial number that is assigned to every agent. The unique linear order of the communication ID numbers reflects the unique order of positioning of the agents on a one-lane road. Any set of agents can compare their serial communication ID numbers and conclude the same unique ordering of their positions on the road.

The flow diagram for rendezvous protocols of a *join* of agents A and B in Figure 3.6 is presented in Figures 3.7 to 3.9. The protocols are designed such that at any moment any agent is aware of the communication address and status of its *front* and *behind* agents. While B is performing a join, C is aware of it and sets the status of its *front* agent to *busy*. Once the action is done, A and C will update their related information. The flow diagrams for *split* are similar.

3.4.5 Simulation and Verification

We focus on the structure of the protocols and verify their completeness and logical consistency. We present an overall description of the protocols in a modeling language and ignore issues such as the message format, encoding, etc. The protocols are verified assuming error-free communication channels.

PROMELA is the modeling language we use to express the address resolution protocols. It is a set of notations for the specification and verification of procedure rules [27]. PROMELA is simple yet sufficiently powerful to represent our protocols. It allows for concurrent processes, such as rendezvous communications between different vehicles.

A brief description of PROMELA is useful for understanding the protocols. There are three specific types of objects in the language: *processes*, *message channels* and *state variables* [27]. We have used a process to describe an agent's addresser. Message channels could be defined locally or globally and could store messages. In rendezvous communication that is used here, there are no stored messages in the channels. The configuration model is

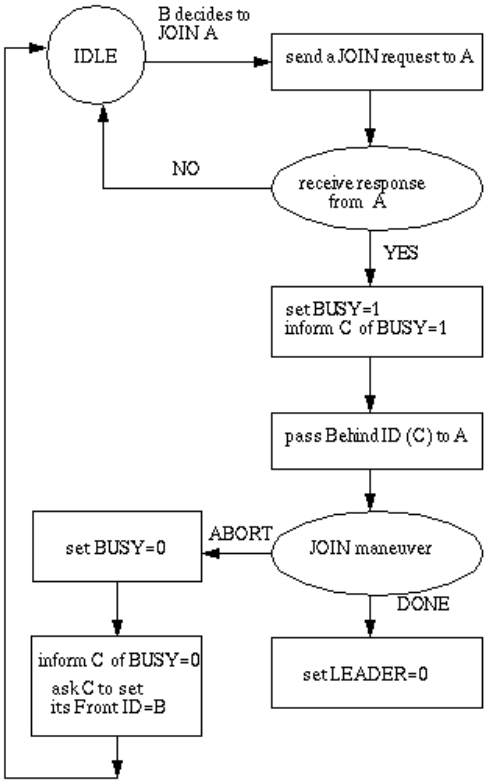

Figure 3.7: *Agent B wants to JOIN agent A, B requests a JOIN with A.*

described by state variables.

What is important in PROMELA is the executability of statements. There is no difference between conditions and statements. The execution of a statement is conditional on its *executability*. Executability is the basic means of synchronization. A process can wait for an event to happen by waiting for a statement to become executable [27]. The *atomic* command is for running concurrent processes and forces the addressers to be able to interleave and execute their statements together. For example, every addresser is modeled by a process and an addresser is sending a *confirm* message to its neighbor and waiting to receive an *acknowledge*. The next executable step will be receiving the *acknowledge*, and, if

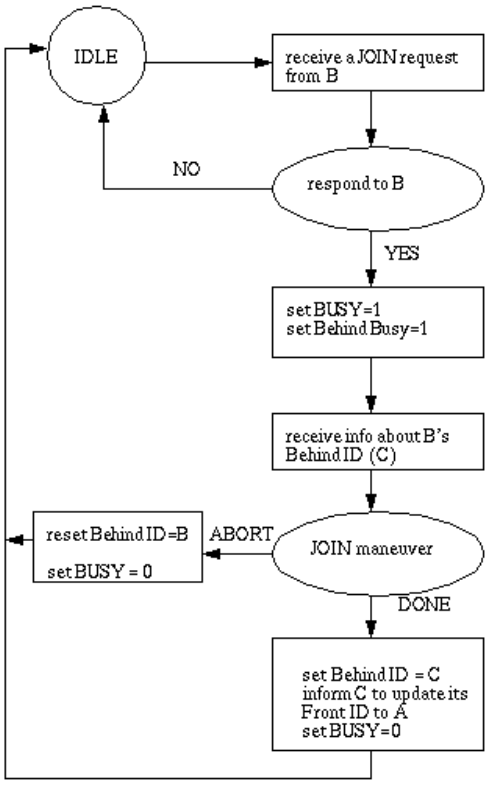

Figure 3.8: *Agent B wants to JOIN agent A, A responds to a JOIN request by B.*

it is not received, its process is not executable and would stop if receiving the *acknowledge* is its only option to proceed. In our protocols there is another option, *query*, which, in the case of losing a neighbor to the far distance, allows the addresser to query its neighbor.

Absolute time is not defined in PROMELA, the main point is the relative order of steps. Statements are executed one at a time and there is no parallel execution of separate processes by the software tool.

Verification of the proposed protocols have been done by SPIN which is short for *simple PROMELA interpreter* [27]. Spin is a tool for analyzing the logical consistency of concurrent systems, specifically of data communication protocols [28]. It takes a model system specified in PROMELA and can either perform random simulations (Figures 3.10 to 3.12) or verify the system's correctness properties as specified by the protocol designer. It is capable of searching for the absence of deadlocks, unspecified receptions, unexecutable codes, unreachable states, and non-progress cycles or live-locks. The verification results can be viewed in an output file, as in Figures 3.13 and 3.14. Xspin is the window-based interactive version of spin and its version 2.9.7 has been used for this project.

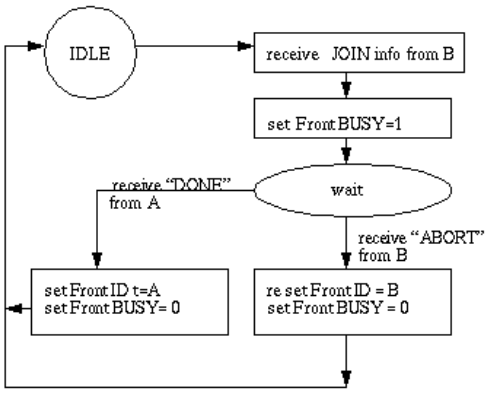

Figure 3.9: *Agent B wants to JOIN agent A, C is behind B, and its Front ID is affected by the JOIN action.*

Simulation has been used in preliminary debugging of the protocols. Also, the simulation output provides an illustration of the protocols function as depicted in Figures 3.10 to 3.12. The address resolution protocol exchange between agent A of Figure 3.6 and its neighbors is simulated as their relative positions change. The protocol exchange with the *front* neighbor is independent of the communication with *behind* neighbors and vise versa. This simulation concerns the address resolution protocols of A and its *behind* neighbors. Figure 3.10 illustrates the *join* process between A and B, while C is being informed of the change in its *front* neighbor's address and its addresser parameters are updated. B becomes a *Follower* after the action and its *leader* binary parameter turns "0". However, before it loses its access to the WAN communication channel, it informs C about its action so that C updates its addresser parameters.

As shown in Figure 3.11, a *Follower* like vehicle B can request a *split* from its *Leader* agent A. If A is not busy, it would acknowledge the request and inform its *behind* agent about the action to be performed. The *behind* agent, C, then updates its model of the *front* neighbor to busy status. As soon as A acknowledges the request, it informs C of the communication ID of B, and informs B of the communication ID of C. When the actions are done, A starts exchanging *confirm* protocols with B, while B engages in a similar protocol exchange with C. It is important that upon every action, the involved agents inform their neighbors about the changes in the configuration. In this case, A not only initializes B, but also provides communication ID of B to C as its future *front* agent.

Figure 3.12 depicts the query protocol of A to find its *behind* neighbor. Once it finds its *behind* neighbor, it would start a *confirm* protocol exchange with it.

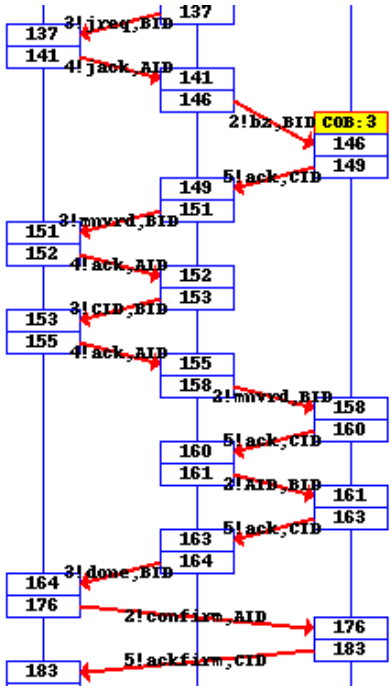

Figure 3.10: *Agent A, left, performs a* join *with B, center, and agent C, right, is informed by both A and B about their action.*

3.4.6 Results

Informally, protocol safety refers to avoiding bad states that should never be reached. In other words, it refers to the correctness of the protocol. Simple propositions can make correctness claims for PROMELA model variables, execution processes, and the contents of message channels. We claim the following.

Once an action happens, the addressers of all neighbors, including self, are aware of the change in the network configuration and reflect it in their model parameters.

Specifically, the assertion is that a change in the network configuration is correctly reflected in the configuration model of the agents.

Temporal claims specify an ordering of propositions. Within one process, the order

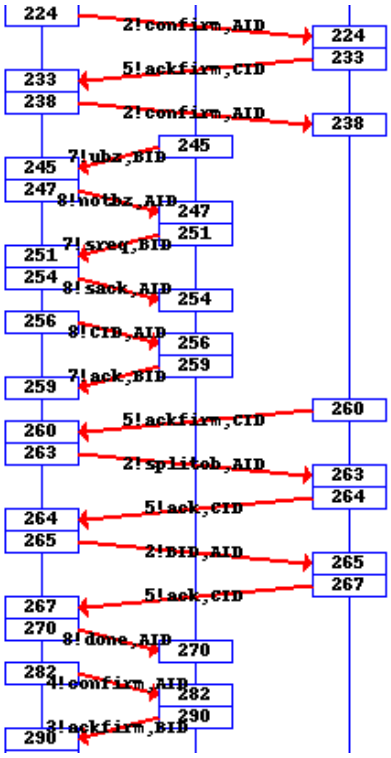

Figure 3.11: *B, center, asks for* split *from A, left.*

of statements shows the order of their executions, while in concurrent processes there is no guarantee about the relative speed of different processes. However, rendezvous communication synchronizes the communication among different agents, and we can verify ordering of propositions in different agents. Again, since we are not allowed to make any assumptions about absolute time, the only valid interpretation of the word "after" in PROMELA is *eventually* after.

We have used temporal claims to verify the functionality or *fairness* of the protocols, e.g., once an agent asks for *join*, its request will *eventually* be granted. This verification along with no error result of the safety statements shows that *changes do happen and are reflected in the models by the addressers.* The correctness claims and the temporal claims

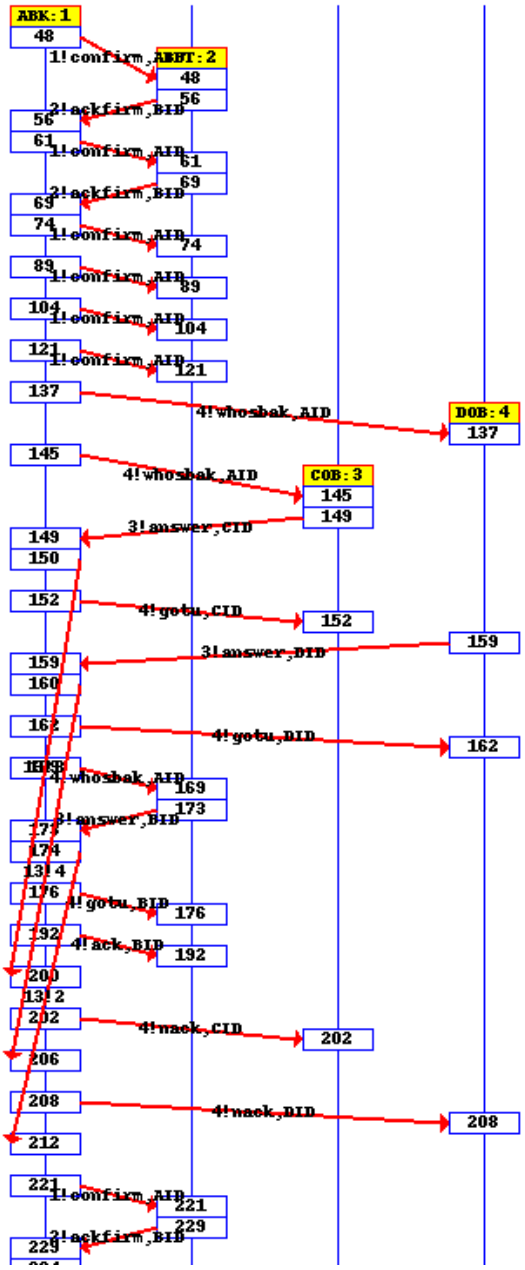

Figure 3.12: *Agent A, left, loses B, second from left, as its neighbor and queries around. B, C, and D, on the right, respond. A successfully finds the nearest neighbor, B.*

can be viewed in [29].

Figures 3.13 and 3.14 show the result of the *safety claim* and the *temporal claim* verification, respectively.

```
(Spin Version 2.9.7 -- 18 April 1997)
        + Partial Order Reduction

Full statespace search for:
        never-claim             - (not selected)
        assertion violations    +
        cycle checks            - (disabled by -DSAFETY)
        invalid endstates       +

State-vector 160 byte, depth reached 23215, errors: 0
  108332 states, stored
   25012 states, matched
  133344 transitions (= stored+matched)
       2 atomic steps
hash conflicts: 12036 (resolved)
(max size 2^23 states)

Stats on memory usage (in Megabytes):
18.200  equivalent memory usage for states (stored*(State-vector + overhead))
        compressed State-vector = 113 byte + 8 byte overhead
13.083  actual memory usage for states (compression: 71.89%)
33.554  +memory used for hash-table (-w23)
2.400   +memory used for DFS stack (-m100000)
1.133   +memory used for other data structures
50.086  =total actual memory usage
```

Figure 3.13: *Verification Output: Safety Claims*

```
(Spin Version 2.9.7 -- 18 April 1997)

Full statespace search for:
        never-claim             +
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness enabled)
        invalid endstates       - (disabled by never-claim)

State-vector 172 byte, depth reached 4975, errors: 0
   19475 states, stored
    7925 states, matched
   27400 transitions (= stored+matched)
       2 atomic steps
hash conflicts: 11444 (resolved)
(max size 2^19 states)

Stats on memory usage (in Megabytes):
3.506   equivalent memory usage for states (stored*(State-vector + overhead))
        compressed State-vector = 129 byte + 8 byte overhead
2.665   actual memory usage for states (compression: 76.02%)
2.097   +memory used for hash-table (-w19)
0.240   +memory used for DFS stack (-m10000)
0.284   +memory used for other data structures
5.204   =total actual memory usage
```

Figure 3.14: *Verification Output: Temporal Claims*

3.5 Address Resolution in Multi-Lane AHS

The address resolution scheme in a one-lane AHS cannot be used for a multi-lane AHS because the communication networks in one-lane and multi-lane AHS systems have different topologies. Although the network topology changes in a one-lane AHS, the relative positions of vehicles do not change with vehicle actions. Therefore, the network that is formed by initialization can facilitate the updating process. In a multi-lane AHS, however, the relative ordering of vehicles in a lane varies due to lane-change actions. Moreover, the relative positions of vehicles in different lanes vary due to different lane velocities. Therefore, vehicles cannot rely on the information provided by initialization.

We take advantage of the technical capabilities of the multi-lane AHS infrastructure and propose a feasible communication scheme that facilitates address resolution. We first propose an address resolution model for a multi-lane AHS. Then, we overview the design specifications of AHS communication network and the technical capabilities of AHS infrastructure. Finally, we present a solution to the address resolution problem for multi-lane AHS that takes advantage of both design specifications and AHS infrastructure capabilities.

3.5.1 Address Resolution Model

A vehicle in a multi-lane AHS needs a configuration model of its neighbors and the binding $< relative\ position, ID >$ for every neighbor in real-time. In a multi-lane AHS, absolute position information is necessary for address resolution as suggested by Puri et al [25]. Therefore, we need to provide absolute position information to every vehicle. The next chapter suggests a vehicle positioning system for this purpose. Here we assume that every

vehicle can determine its absolute position. The following address resolution models use absolute positioning information. Vehicles broadcast their position along with their communication ID numbers. Therefore, a vehicle A receives the binding $< absolute\ position, ID >$ information from every vehicle in its neighborhood. The assumption is that vehicle A knows its own absolute position, and thus, it can obtain the relative positions of its neighbors and the binding $< relative\ position, ID >$ for them. Consequently, every vehicle can construct a real time configuration model and obtain the communication addresses of its neighbors.

3.5.2 Communication Design Specifications

The communication network of vehicles in AHS must satisfy the following specifications.

Periodic Transmission

The network topology and, hence, vehicle configurations, change over time with vehicle actions. A vehicle needs real time information regarding the communication addresses of its neighbors $< relative\ position, ID >$. Therefore, it should have periodic communication with its neighbors for configuration updates.

Uniform Access Time

Address resolution is in demand by all vehicles. The type and amount of communication messages that need to be exchanged are uniform for all vehicles. Consequently, the address resolution scheme must allow uniform periods of access to the communication channel for all vehicles.

3.5.3 AHS Infrastructure

The AHS infrastructure has technical capabilities that we exploit in our communication network design.

Synchronization

The automated road infrastructure can presumably be augmented by advanced technical equipments. Specifically, we assume that the automated road is equipped with synchronized clocks at the entry points. Vehicles are synchronized to these clocks as they enter the road. Consequently, all vehicles are synchronized together.

The design specifications of periodic transmissions and uniform access time make time division multiple access (TDMA) scheme a good solution candidate with respect to our address resolution model. The problem then is to assign time divisions to the dynamic network of mobile vehicles. We solve this problem by a novel method of communication channel division.

3.5.4 Space-Time Division Multiple Access (STDMA)

We design a multiple access method that although similar to TDMA, it is fundamentally different in terms of time assignment methodology. In TDMA, a time period T is divided into $T_1, \ldots, T_n$ among n users where $T_1 + T_2 + \ldots + T_n = T$. In an ad-hoc network of vehicles, users are not initially known prior to their address resolution, and thus, cannot be assigned time divisions.

In our proposed scheme, we assign a time division to the *space* where the vehicle is located in real time. As the result, the time division that is assigned to a vehicle varies

as the vehicle moves along the road while a space division on the road has a constant time division with respect to the time period T. Specifically, we divide a time period T among space divisions in the following manner. Consider a road segment S of length L. We divide the road segment S into *space divisions* $s_1, \ldots, s_n$ and assign time divisions $T(s_1), \ldots, T(s_n)$ to these space divisions, respectively, where $T(s_1) + \ldots + T(s_n) = T$. Let $\mathcal{L}(s_i)$ denote the length of s_i. The space divisions do not overlap and they cover the entire road, $\mathcal{L}(s_1) + \mathcal{L}(s_2) + \ldots + \mathcal{L}(s_n) = L$. A unique time division $T(s_k)$ is assigned to every space division s_k on the road. A space division may or may not contain a vehicle. We call this scheme *space-time division multiple access (STDMA)* which uses space divisions to allocate time divisions for multiple access communication.

3.5.5 Implementation Issues

Implementing STDMA scheme in AHS requires that a vehicle have the following information.

I Absolute positions of the space divisions on the road.

II The corresponding time division assignment for space divisions.

III The position of the vehicle with respect to the space divisions, and hence, the assigned time division with respect to the time period.

IV The common reference clock time to synchronize clocks and transmit on the assigned time division as in TDMA scheme.

The information I and II regarding space divisions and the related time divisions are assumed to be known *a priori* by all vehicles. However, a vehicle needs *real time* information of III and

IV regarding its current space division and common reference clock time. Real time position information requires a positioning system. We propose a vehicle positioning system for this purpose in the next chapter. The common reference clock time requires synchronization of all vehicle clocks. As noted we assume that the AHS infrastructure capabilities allow vehicle synchronization. This can be achieved for instance by employing a GPS receiver at every entry point on the road to obtain the GPS time. The road then synchronizes vehicle clocks to the GPS clock.

Time and Space Margins

An important design and implementation issue is to guarantee lack of communication interference. In TDMA scheme, time margins are considered to guarantee that consecutive time divisions do not overlap. We need to guarantee that there is no more than one vehicle transmitting at a time. While TDMA scheme assumes time margins to take care of clock mismatch errors, STDMA scheme provides space margins to take care of positioning errors. The longitudinal safety margins in AHS that were discussed in the previous chapter play an instrumental role in providing space margins. The clock mismatch errors also exist and must be considered. However, as we see in the following example, timing errors with GPS technology is relatively negligible in this AHS application.

We provide an example of STDMA scheme on a three-lane AHS to provide more details about the STDMA scheme and space margins.

3.6 STDMA Example

We consider the STDMA scheme for address resolution in in a three-lane AHS. We assume the "worst case" scenario in terms of having the maximum communication demand on the road. We first review some assumptions and design specifications for the worst case of communication demand with respect to address resolution. We then define the space divisions and their corresponding time divisions and discuss the space and time margins. Finally, we discuss the STDMA bandwidth efficiency.

Technical Assumptions

We assume that the AHS infrastructure is equipped with GPS clocks and all vehicles are synchronized to the GPS time.

Assumption 4 *The relative clock accuracy of vehicles is within* 100 *nanoseconds.*

The positioning accuracy of vehicles can be determined by system designers as we explain in the next chapter. Here, we consider a conservative estimate of the positioning system accuracy.

Assumption 5 *Positioning accuracy is* ± 2 *meters.*

Design Specifications

We emphasize again that the WAN communication scheme that was described in Section 3.1 involves the platoon *Leaders*. A *Single* vehicle is also considered a *Leader* because it is active in the WAN communication.

The communication requirement is defined by the regulation layer safety constraint [19, 21, 30].

Specification 1 *Every vehicle broadcasts every* 20 *milliseconds approximately 100 bytes of data.*

In order to avoid communication interference, we allow one platoon *Leader* per space division. We assume the highest demand for communication in the WAN. That is, we assume the maximum road capacity for WAN users. Note that in a given space we have the maximum number of *Leaders* when every *Leader* is *Single*.

Specification 2 *All vehicles are* Single, *i.e., the maximum platoon size is one.*

The road capacity for the WAN users is maximized when the distance between the users is minimized. We assume a very small safety margin as the distance between two contiguous *Leaders*.

Specification 3 *The longitudinal safety margin is* 10 *meters.*

We consider an interaction range for vehicles and assume that a vehicle could interact and coordinate actions with other vehicles that are at most 150 meters away. Therefore, we specify the communication range for vehicles accordingly.

Specification 4 *The communication range of vehicles is* 150 *meters in each direction.*

Two vehicles can transmit simultaneously if their communication ranges do not overlap. In order to secure non-overlapping communication ranges under all conditions, we require 200 meter distance between the communication ranges of simultaneously transmitting vehicles. This requirement and Specification 4 lead to the following specification.

Specification 5 *A time division within a time period can be used simultaneously by two vehicles if they are at least* 500 *meters apart.*

Therefore, we consider the vehicles on a 500 meter segment of the road that share a time period T. We define space divisions in order to allocate time divisions to the vehicles.

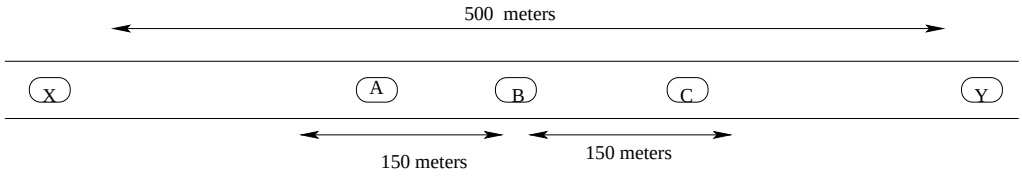

Figure 3.15: *Vehicles A and B and C cannot transmit simultaneously because vehicles A and C are both within the interaction range of B. However, vehicles X and Y can use the same time division for transmission.*

3.6.1 Space Division and Time Allocation

We consider the maximum number of *Leaders* that occupy 500 meters of the road. First, we divide the road into space divisions such that no two *Leaders* can be on the same space division at one time. Then, we divide a time period of 20 milliseconds among these space divisions. This assignment of time divisions to the space divisions can be repeated every 500 meters on the road.

The above assumptions lead to the following calculations for the length of space and time divisions. In each lane of the 500 meter segment of the road there are (500 $meters \div$ 10 $meters$ =) 50 vehicles. In the three lanes, there are (3×50 =) 150 vehicles. These vehicles need unique time divisions within a 20 millisecond period. Therefore, every vehicle has a time division of ($20 millisecond \div 150$ =) 130 microseconds which is sufficient for transmitting 100 bytes of data by the currently available 2.4 GHz wireless technology.

Figure 3.16 depicts the related time allocations for the space divisions on the road.

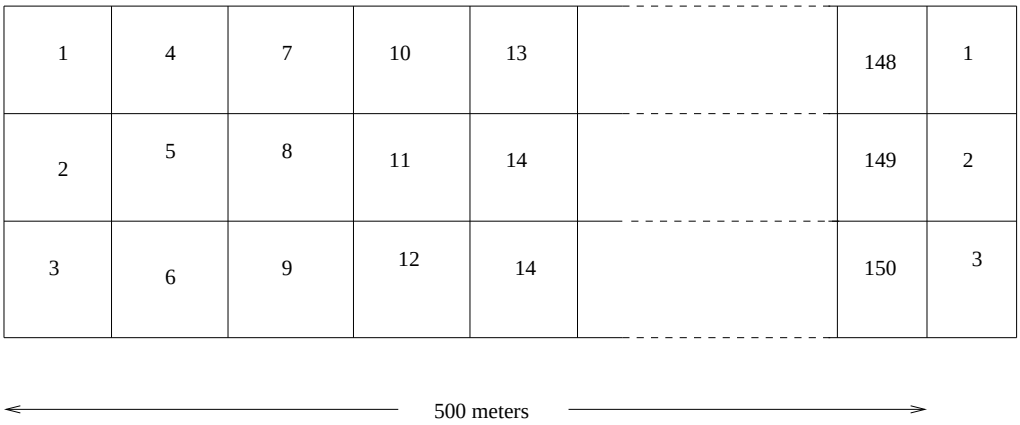

Figure 3.16: *The space divisions on the road that correspond to the time divisions in STDMA scheme.*

3.6.2 Sources of Error

The potential problems with the scheme presented in Figure 3.16 are possible clock mismatch and positioning errors.

Clock Mismatch Errors

The timing error that causes the relative clock mismatch between different vehicles may cause overlapping transmissions. When vehicles are moving at the speed of 40 meter/second, the 100 nanosecond relative clock error results in 0.004 millimeter of distance. We can consider transmission time margins similar to TDMA time margins. Alternatively, we can add 0.004 millimeter to the space margin that is discussed below.

Positioning Errors

The positioning error of vehicles may also cause two vehicles to assume they are in the same space division and transmit simultaneously. In order to ensure that no two contiguous vehicles use the same time division for transmission, we provide space margins in our STDMA scheme.

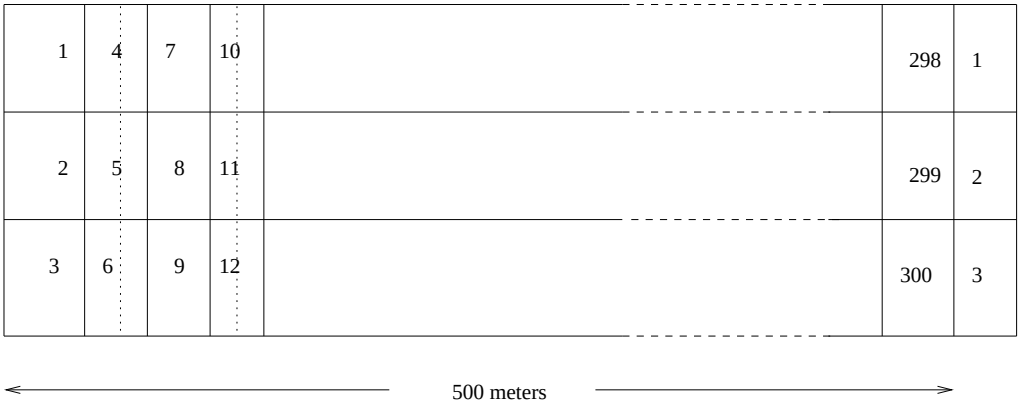

Figure 3.17: *By assuming margins for the space division we guarantee that vehicle positioning error will not cause simultaneous transmissions. (The dotted lines show the original space divisions.)*

3.6.3 Space Margins

Figure 3.17 illustrates the space divisions with margins. Unlike time margins in TDMA, the space margins in STDMA are not idle, but provide active communication "buffer" zones. That is, we divide the road into smaller divisions where all divisions are active in the communication scheme and are assigned time divisions. We add the "margins" by reducing the length of space divisions from 10 meter to 5 meter. The *longitudinal safety margins* specifies a minimum distance of 10 meters between two vehicles. Therefore, the worst case positioning errors could not cause simultaneous transmissions within a commu-

nication range. Figure 3.18 depicts two vehicles A and B with 10 meter distance. Suppose A has $+2$ meter positioning error while B has a positioning error of -2 meters. They still would assume 6 meters distance from each other and would consider different space divisions for their transmission times. The space margins make this scheme robust even in the presence of the positioning errors by vehicles.

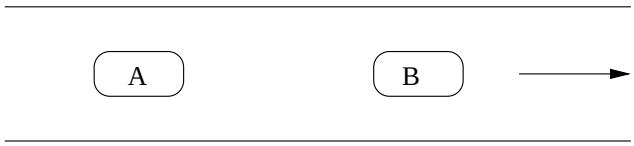

Figure 3.18: *Vehicle A has positive positioning error where B has negative positioning error where the direction of motion and positioning measurement is shown by the arrow.*

3.6.4 Bandwidth Efficiency

We considered the "worst case" scenario of communication demand as the case of having the maximum number of WAN users on the road. However, this case has the maximum bandwidth efficiency. When the number of *Leaders* on the road is less than the maximum road capacity for WAN users, then there are space divisions that are not occupied by vehicles and their assigned time divisions would be idle.

The efficiency of this scheme depends on the number of WAN users on the road. The result is that the bandwidth efficiency of STDMA scheme is variable and proportional to the number of users. This "variable efficiency" is the price we pay for having a robust ad-hoc network with self-start address resolution in the dynamic mobile network of automated vehicles.

We mentioned above that a positioning system is necessary for implementing

STDMA scheme. In the next chapter we propose a positioning system that is suitable for the AHS and provides sufficient accuracy for STDMA application.

Chapter 4

Communication Requirement: Vehicle Positioning System

We propose a novel method for position location by automated vehicles in AHS that provides the basic information requirement for communication address resolution. The method employs magnetic signals to transmit pseudo-noise codes. Magnetic markers are provisioned on automated roads for lateral control of vehicles within a lane [31]. A vehicle has a magnetometer to follow a sequence of markers centered on the lane. We propose a technology for vehicle location that takes advantage of this AHS infrastructure. The magnetic markers are binary coded using their dual polarity. The sequence of the binary magnetic markers at each lane is coded to yield a pseudo-noise signal that is unique to that lane in the AHS network. The phase of the pseudo-noise signal represents the receiver's range from a reference point on the lane, say, the beginning of the sequence on the lane. A vehicle resolves its absolute position on the road by estimating the signal phase. The signal

properties insure accurate phase estimation. The advantage of our method over other navigation technologies is that it provides continuous accurate positioning. Moreover, vehicle positioning is obtained in real-time with little data processing. Radio navigation technologies fail to promise continuous accuracy due to signal fading and multi-path problems.

We first provide a brief background on positioning systems and an overview of the specifications for positioning system in AHS. We then discuss maximal length PN sequences and explain our system of positioning by magnetic maximal length codes. We discuss system components and design issues and present an analysis of error and cost. Finally, we provide suggestion for improvement.

4.1 Background

Positioning systems such as GPS use pseudo-noise (PN) signals for range measurement [32, 33, 34, 35, 36, 37, 38, 39]. PN signals provide excellent accuracy in range measurement due to their correlation and autocorrelation properties. However, a system such as GPS may not be the right solution for automated vehicles because of the errors such as multi-path that are associated with radio systems [40, 32]. Accuracy in positioning is important for AHS and an automated vehicle at any point must be aware of its degree of location accuracy in position location to choose its control policy accordingly.

Magnetic markers are provisioned on automated roads for lateral control of vehicles within a lane. Suppose that the markers carry a PN code and an automated vehicle knows the code layout on the road with respect to the physical location of every binary chip. The vehicle has a replica of the signal and measures the phase of the received signal with

respect to the replica and infers its position on the road. Because of the nature of our system infrastructure, the probability of error is negligible compared with radio navigation systems. However, errors do exist and an attractive feature of our method is its error detection capability. Errors can be detected and corrected shortly after they occur and the error detection and correction delays are subject to design. Error bounds can also be specified with respect to the signal code design. Unlike radio signals, magnetic codes do not interfere, and multi-path is not an issue. However, noise is present and will be discussed in our error analysis.

4.2 System Specification

A positioning system for automated vehicles in AHS must meet the following specifications.

Independence: The system should maintain its integrity independent of other components of the AHS system.

High Precision and Accuracy: Increased precision allows for reduced space taken by vehicle maneuvers. Thus, throughput of the automated road is improved as the positioning system becomes more accurate.

Reliability: Functionality, precision, and accuracy of the system must be independent of the physical environment, weather condition, surrounding walls (tunnels), and road visibility.

Fault Awareness: The system should allow an estimate of the positioning error in order to detect faults. This allows vehicles to avoid or abort unsafe maneuvers.

Error Bounds: Given an AHS design, the system level requirements of the design will impose constraints on a feasible positioning system and its error bounds that can be tolerated.

Economy: Feasibility of the positioning system well depends on its cost.

The proposed system meets the AHS requirements and the system is economically feasible because it takes advantage of the infrastructure and characteristics of the automated roads. These characteristics are not met by the current radio navigation systems including GPS.

4.3 Maximal Sequence Codes

Code sequences generated by shift registers are popular because a small number of shift registers can generate relatively long codes. In a ranging system the type of code used, its length, and its bit rate set bounds on the capability of the system that can be changed only by changing the code. The longest codes that can be generated by a given shift register are called maximal sequence codes. The bits of PN sequences are called chips to underscore that these codes do not carry data. In binary shift register sequence generators, the maximum length sequence has $2^n - 1$ chips, where n is the number of stages in the shift register. There is a specific sequence for a shift register, depending on its feedback configuration. Even though the sequences have some randomness properties, the maximal linear sequences are deterministic as sequences repeat at intervals of $2^n - 1$ chips. Each repetition exhibits the same one-zero distribution. The statistical distribution of ones and zeros is well defined and always the same. As the window of observed values within a

period increases, the randomness of one-zero distribution decreases. Maximal sequence codes have the following properties that allows our system to have accurate positioning and error detection and correction capabilities [41, 32].

Property 1

An important property of maximal codes is the predictability of codes. Knowing n, the number of shift registers, and $2n$ consecutive chips of the code, one can predict the entire length of $2^n - 1$-chip code. This is because the feedback configuration can be realized by solving equations of the form $b_i = \sum a_j b_{i-j}$. Since there are n stages and each could be involved in the feedback configuration, n equations are needed.

$b_{n+1} = a_1 b_n + a_2 b_{n-1} + \ldots + a_n b_1$

$b_{n+2} = a_1 b_{n+1} + a_2 b_n + \ldots + a_n b_2$

$\ldots$

$b_{n+n} = a_1 b_{n+(n+1)} a_2 b_{n+(n-2)} + \ldots + a_n b_n$

The equations have overlapping blocks of $n + 1$ consecutive chips from the sequence, and n equations are needed, which sets the required number of known chips to $2n$. Thus, the feedback structure of an n-stage shift register can uniquely be determined from $2n$ chips. For example, having 20 chips of a 10-stage shift register, one could predict the entire $210 - 1 = 1023$ chips of the code. This property, a disadvantage for secure spread spectrum communication, is used to great advantage in our system.

Property 2

Autocorrelation of a maximal code which is the degree of correspondence between a code and a phase-shift replica of itself is such that for all values of phase shift the correlation value is -1, except for the $pm1$ chip phase-shift, in which correlation varies linearly from -1 value to a maximum $2^n - 1$. The autocorrelation function $R(\tau) = \sum_{i=1}^{n} b_i b_{i+\tau}$ is depicted in Figure 4.1 where $b_i b_{i+\tau}$ is equal to 1 if $b_i = b_{i+\tau}$, and -1 if $b_i \neq b_{i+\tau}$. The plot shows the number of agreements minus disagreements over the length of the two codes being compared, as the codes assume every shift number in the set of shifts of interest. Such a plot is two-valued, with a peak only at the zero shift point. This is an invaluable property because it allows the receiver to discriminate between signals on a yes-no basis.

When the period $2^n - 1$ is large, the full period correlation loses some of its value as a design parameter. Correlation calculations in this case typically are carried over blocks of K chips, where K is larger than n, the number of stages of the code generator, and smaller than the code period, $2^n - 1$. A more appropriate statistic for study in this case is the partial autocorrelation defined as $R(\tau, K) = \sum_{i=1}^{K} b_i b_{i+\tau}$. This computes the cross-correlation between two blocks of K symbols, one block located symbols from the other.

Property 3

The number of ones and zeros are equal in a sequence. For example, a 127-chip code has 64 ones and 63 zeros. The number of ones in any linear maximal code is $2^n/2 = 2^{n-1}$, and the number of zeros is $2^n/2 - 1 = 2^{n-1} - 1$, where n is the number of stages in the code generator, and the code length is $2^n - 1$ chips. This randomness property allows for a low

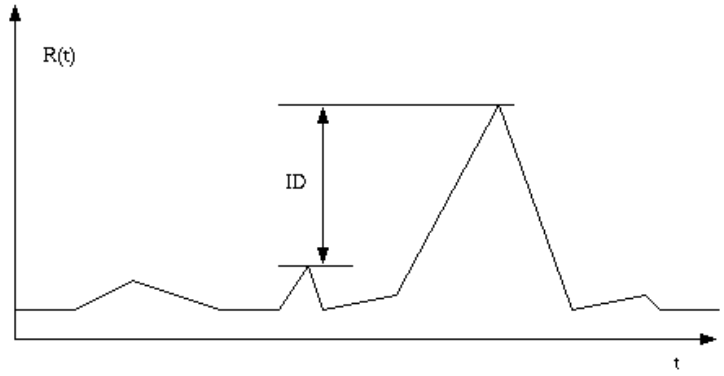

Figure 4.1: *Index of discrimination (ID) denotes the difference in correlation between a fully correlated, i.e., perfectly synchronized, code and the peak of minor autocorrelation or of cross-correlations. Maximal sequence codes have a high ID which makes them suitable for positioning systems.*

value of autocorrelation of a PN code with its phase-shifted replica. This property is used in phase measurement.

Property 4

A run is a finite sequence of repeating ones or zeros. Relative positions of the runs vary from code sequence to code sequence, but the number of each run length does not. Freymodsson has shown that there are exactly $2^{n-(m+2)}$ runs of either ones or zeros of length m in every maximal code sequence [41]. The exception is that there is only one run containing n ones and one containing $n-1$ zeros. There are no runs of zeros of length n or of ones of length $n-1$.

Property 5

A modulo-2 addition of a maximal linear code with a phase-shifted replica of itself results in another replica with a phase shift different from either of the originals.

4.4 Positioning

We use the properties of maximal sequence codes in positioning by magnetic markers. The process of position location is illustrated in Figure 4.2.

Assume that the magnetic markers in each lane are coded with a unique maximal code with a linear span of n. By property 1, reading a $2n$-chip segment of a maximal code is sufficient to predict the entire period. The vehicle can figure out the feedback configuration of the code, and build the related feedback configuration and generate a replica of the code.

Property 2 allows perfect synchronization of the vehicle replica and the received signal. This enables the vehicle to estimate the exact phase of the magnetic PN code and figure out its absolute position using the location map of the code layout.

Property 4 guarantees that a $2n$-chip block uniquely determines the code phase. The proof is by contradiction. Suppose a $2n$-chip block is repeated within one period. Shift a replica of the code, and add it to itself such that the two similar blocks add. By property 5, this would result in another replica with a phase shift different from either of the originals. This would result in a $2n$-long run of zeros while the maximum run length for zeros is $n-1$ by property 3.

After synchronization, position updating is done by tracking the magnetic signal and checking it against the replica. Tracking is done by predictable partial autocorrelation. The code is predictable by the receiver, and it is partially being autocorrelated with its replica, i.e., over a partial period of $2n$ chips. As long as the autocorrelation function results in the maximum value, which is equal to the number of received chips, the reading is assumed to be correct.

Tracking allows the vehicle to detect the errors in reading the magnetic code. For example, assume the next chip in the replica is "1", but the next reading of the received signal is a "0", or vice versa. In this case an error is detected, and the receiver must read a sufficient number of chips correctly before it can assert a re-synchronization. Note that no false errors are detected. However, error detection may not happen in real time. For example, assume that the first "1" in a run of five "1"s is missed by the receiver. The missed "1" is shown by "∅" below.

$received - code \qquad \ldots 0\emptyset 111100 \ldots$

$replica - code \qquad \ldots 01111100 \ldots$

The receiver will not know the missed reading until it reaches the end of the run in the received signal, and starts receiving zeros. The error in positioning is only one magnetic spacing (chip), but it takes a five magnetic spacing trip for the receiver to detect the error.

Figure 4.2 shows that in case of an error, $2n$ chips must be read for re-synchronization. The optimum number of chips depends on the specific code and could be smaller than $2n$.

4.5 System Components and Design

The system elements for positioning include both hardware and software elements. Some of the software components can be built as more efficient and faster hardware elements in actual implementation of the design. For example, most of the synchronization and phase tracking processes can be done by hardware.

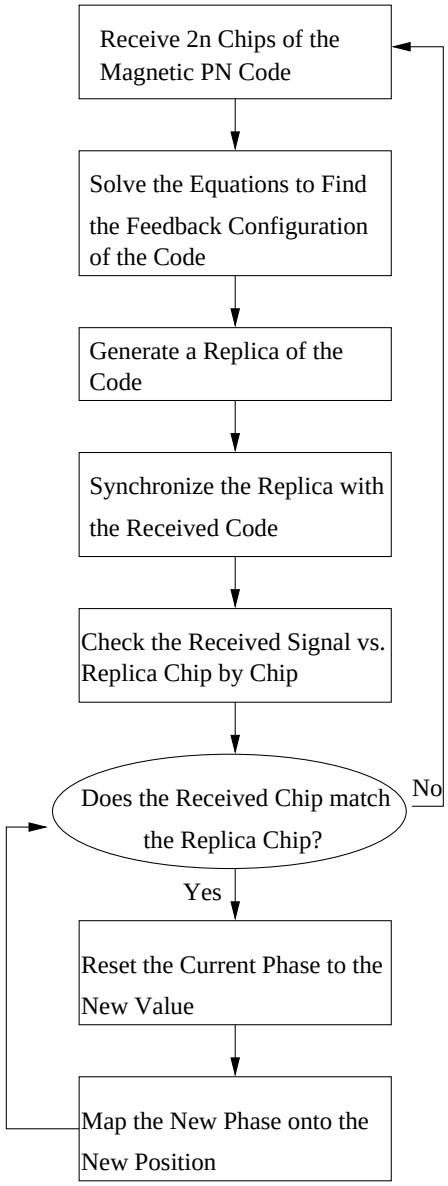

Figure 4.2: *Flowdiagram for magnetic positioning system*

4.5.1 Infrastructure

Magnetic markers are used in the design of AHS for lateral control of automated vehicles and their binary polarity is used to code information about the road curvature and similar data [31]. We are proposing a modification to the existing design, i.e., coding magnetic markers by maximal sequences. The distance between magnets sets the accuracy of positioning. The magnet spacing could be set to optimize precision versus cost.

4.5.2 Software

Consider the set of magnetic sequences $\mathcal{S}$ that are used for positioning by magnetic markers, and the set of road maps $\mathcal{L}$. There must be a one-to-one mapping $\theta : \mathcal{S} \rightarrow \mathcal{L}$ such that a sequence of magnetic markers has a unique physical representation . Suppose there are N automated roads in an area, and each has M automated lanes in both directions. This makes a total of magnetic codes and requires the same number of physical interpretations. Thus, a vehicle that knows the phase of the magnetic signal can obtain its position on the road. Currently, the markers are designed to provide specific information such as road curvature [31]. A mapping that provides information about the absolute position of a vehicle on the road can be linked to other data about the road, travel information, etc.

4.5.3 Code Selection

A design issue in phase synchronization is that the code must be long enough to avoid any ambiguity in resolving the phase of the code, i.e., the period of the magnetic sequence must be longer than the road. After synchronization, the position update during the motion must be tracked by predictable partial autocorrelation. Synchronization is

possible by reading $2n$ chips where we can choose a set of codes of equal length of $2^n - 1$. Assume that the distance between two consecutive magnetic markers is one meter. A vehicle must travel $2n$ meters before it can initialize its position and start updating it by tracking the signal. If the linear span of the code, n, is too large, it will take a long ($2n$-chip) travel distance and computation time for a vehicle to find its location. If the code period is so short that the code period is shorter than the road length, it could cause ambiguity about the number of code periods that are repeated on the road. When a vehicle enters the road, or changes lane, it must be able to initialize its position without ambiguity.

Another consideration for code length is the number of different codes that we would like to have for synchronization. The maximum number of different codes of length $2^n - 1$ is limited by $(pf_1 - 1)(pf_2 - 1)\ldots/2n$, where pf_1, pf_2 ,... are the prime factors of $2^n - 1$ [41].

Example 5 *Let* $n = 21$. *The prime factors of* $2^{21} - 1 = 2,097,151$ *are* 7, 7, 127, *and* 337. *There are unique codes of the length* $2^{21} - 1 = 2,097,151$ *meters, which is well beyond the length of the road. The feedback configuration of the road can be computed from* $2n$ *chips, and hence, the entire code can be predicted. A vehicle has to travel* $2n$ *chips, or* 42 *meters, before identifying the code.*

In case of branching roads, such as in Figure 4.3, each road could have its own code, or the consecutive segments $R1$, $R3$, and $R4$ could have one code, and $R2$ and $R5$ have two other different codes. Assume a vehicle is moving from $R2$ to $R3$ where the code sequence changes at point A. If the receiver is not aware of the code change, it will detect errors in partial autocorrelation and will try to re-synchronize with the code of $R2$, whereas

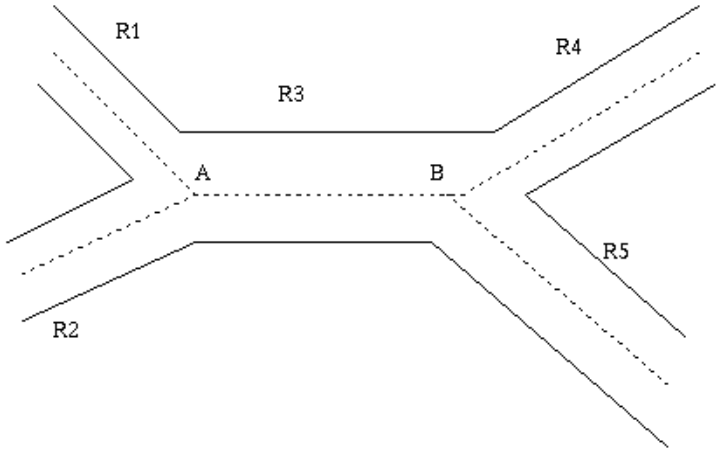

Figure 4.3: *A branching road requires special considerations in PN code design.*

it really needs to identify code $R3$ and synchronize with it. In this case, the coding at point A must be such that the starting code of $R3$ has a low partial correlation with the replica of $R2$. Hence, the receiver detects an error as soon as it starts reading $R3$ code, and will identify the new code and perform synchronization. Alternatively, the software could provide road information such that a vehicle that is going from $R2$ to $R3$ is aware of the road change and knows the new code and will only perform a synchronization upon entering $R3$.

4.6 Error Analysis

It is important for vehicles to choose their control policy based on the information accuracy. In case accurate information is not available, vehicles should be aware of the degree of available accuracy and perform accordingly. For example, if the safety space gap around a vehicle for lane change maneuver is set at 30 meters, and a vehicle's degree of

accuracy is ± 1 meter, it must keep a distance of 31 meters to count for a possible error.

The error analysis of positioning with magnetic markers is conditioned on the fact that the lateral control of vehicle is working properly. This implies that the error cannot exceed certain amount specified by lateral control regulation. The magnetic markers are the essence of lateral control and the error probability p is very small, currently its specification is in the order of 10^{-3}.

The error in positioning by a magnetic signal is due to error in the signal. The error could be received due to (i) missing a marker, (ii) misreading a marker (reading a "0" as a "1" or vice versa), or (iii) reading magnetic noise that is not a marker and was not meant to be read.

4.6.1 Missing a Marker

There are exactly $2n - (r + 2)$ runs of length r for both "1"s and "0"s in every maximal code sequence except that there is only one run containing n ones and one containing $n - 1$ zeros [41]. There are no runs of zeros of length n or ones of length $n - 1$. We consider cases of (a) missing one marker, and (b) missing two or more markers where we assume that the distance between two consecutive markers is one meter.

We first consider case (a). An error that happens during a run length will be detected at the beginning of the next run. If the last chip of a run is missing, reading the first chip of the next run will immediately reveal that there is a missing chip, i.e., an error has occurred. Otherwise, the error would drift and a vehicle will not notice the miss until it starts reading the next run. There are a total of $2^n/2^1$ chips at the end of a run, $2^n/2^2$ chips next to the end of a run,...., and $2^n/2^m$ chips are in the position of $m-1$ chips before the last

chip in a run. Assume uniform probability of missing a chip in a code period. Then, with probability of 1/2 when a chip is missing, the error will be carried for one meter travel, with probability of $1/2^2$ the error will be carried for two meters, and with probability of $1/2^{2m}$, the error will be carried for m meters. The expected number of meters that a one meter error is carried before the error is detected is 2 meters, $1 \times 1/2 + 2 \times 1/2^2 + \ldots + n \times 1/2^n \leq 2$. The worst case of error drift is when a vehicle travels n meters before it detects an error of one meter it has been carrying in its position because by property 4, the maximum run length is n chips.

We now consider case (b) of missing two or more markers. Similar to case (a), the expected number of meters that a car travels before an error is detected can be calculated. However, in this case we need to know the exact code to estimate the number of magnets a vehicle reads before it detects an error.

4.6.2 Misreading a Marker

This will immediately result in error detection as the partial autocorrelation function will be reduced from the expected maximum value. A vehicle then can re-synchronize its receiver, and update its position. The probability of misreading a marker can be significantly reduced by improved software algorithms or implementing hardware.

4.6.3 Magnetic Noise

This is due to presence of a magnetic field, i.e., a magnetic object on the road. Since magnetic field is inversely proportional to the third power of distance, the source of magnetic noise must be physically very close to the markers on the road. It is likely that

such sources of noise are detected and removed before the road is utilized. However, we must count for the low probability of confronting such noise. When noise is introduced, assume that lateral control remains in place and vehicle continues to read the magnetic markers. Furthermore, assume that a noise has equal probability of being a zero or one. If the polarity of the noise chip matches the polarity of the current run, it will not be detected until the end of the run. This is similar to error detection in case (a). If a noise with opposite polarity of the current run happens, it will be immediately detected, similar to the case (ii).

4.6.4 Error Detection and Correction

Assuming one meter distance between markers, and the error probability p in reading markers, the amount of error is one meter with probability of p. Assume uniform distribution of error type (i) over a code period. Errors will not be drifted and accumulated beyond one run. Predictable partial autocorrelation is used in obtaining the phase of the code, and respectively, the position of the vehicle. Perfect partial autocorrelation can be achieved when the received signal and the synchronized replica match, i.e., $R(K, \tau) = K$ where $K < 2n$ is the number of received chips that the receiver can hold in the buffer. If a vehicle holds K received chips in its buffer, but $R(K, \tau) < K$, then an error has happened. However, an error may not be detected instantly. As an example, consider the following portion of a code

...011111000....

The vehicle is reading the run of ones and is checking the received chips against its own replica of the signal, i.e., is calculating the partial autocorrelation function. As long as

$R(K, \tau) = K$, where K is the number of received chips in the buffer, it will not detect an error. Suppose the first "1" of the run is missing, and the vehicle starts reading the run with the second "1". The missed marker is shown by "∅" below.

...0∅1111000....

The vehicle will not detect the missed chip until it reaches the end of the run, i.e., starts reading the run of "0"s when it realizes that the autocorrelation is not maximized. The error is detected at this point, and will be corrected by re-synchronization. Thus, an error that happens within one run, will be detected and corrected at the next run. Consequently, probability of having an error of two meters is the probability of two errors occurring within a run which is $\begin{pmatrix} n \\ \\ 2 \end{pmatrix} p^2(1-p)^{n-2}$ because the maximum run length is n.

The errors do not accumulate, and the beginning of each run confirms lack or existence of error in the previous run. In other words, an error could be carried only during one run without being detected. This may be another consideration in choosing n, the linear span of the code, since the maximum run length within a period is equal to n. Furthermore, during each run, a vehicle can have a maximum of $h-j$ chip errors, where h is the length of the run, and j is the number of received chips in the run. Recall that a quarter of chips in one period of the code have a run length of 1, which improves the accuracy of the system.

4.7 Cost Analysis

The cost of positioning by magnetic PN signals can be divided into implementation, maintenance, and receiver operation costs.

Implementation: The implementation cost includes infrastructure, software, and

noise reduction. Recall that the magnetic markers are an element of the AHS design, and their installation cost is already included in the AHS. However, the exact polarity and location of markers must be recorded for the purpose of mapping the signal phase to absolute position.

The software includes two items. First, a code identification algorithm to resolve the feedback configuration of each code from $2n$ chip that must be implemented in the receiver. Similarly, algorithms for synchronization, tracking, and error detection and correction must be implemented. Second, a one-to-one mapping of each magnetic code sequence phase to the absolute position on the road, $\mathcal{S} \rightarrow \mathcal{L}$, must be provided to each vehicle that uses the AHS network.

In order to reduce the error in the positioning system, it is important to check the automated roads for possible sources of magnetic noise during implementation. This may include costs such as removing a magnetic core from the road asphalt, or displacing some metallic borders around bridges, etc. Alternatively, such permanent sources of error could be included in the magnetic code and the related software. Noise reduction is equally important for lateral control of vehicles, and its cost may be a part of already predicted costs of AHS.

Maintenance

Periodic maintenance of the roads is necessary to ensure correct polarity of magnetic markers. The roads must also be checked for any new sources of magnetic noise.

4.7.1 Receiver Operation

A receiver's operation includes code identification, synchronization, and tracking. In the following, we consider the computation cost for each operation.

Every time a vehicle enters an automated road or changes lanes, it must first identify the code by finding the feedback configuration of the code that it is receiving. Recall the set of n equations that reading $2n$ consecutive chips provide. The calculations for finding the feedback configuration from n equations involves checking the received signal chips against the n equations (1) and finding the feedback configuration through elimination process [41]. The actual number of possibilities for feedback configurations in an automated road network is the same as the number of codes used or the number of automated lanes, say . The set of possible codes can be initially provided to a vehicle. A vehicle needs to try each configuration for n equations in the worst case of search. This amounts to arithmetic operations. A more sophisticated algorithm could decrease the computation complexity.

Next, the phase of the code must be determined through synchronization. First, a replica of the code is generated using its feedback configuration. By maximizing autocorrelation of the replica with the received chips, the phase can be determined. Maximizing autocorrelation function over $2^n - 1$ chips requires $2^n - 1$ calculations. This complexity can be significantly reduced by providing some information that narrows the window of autocorrelation from the entire code period to a relatively small partial period of the code. For example, a vehicle that already has the absolute position of the current lane and wants to change lanes, can use the inverse mapping : and use its current location to estimate the code phase in the next lane and narrow the autocorrelation window in the next lane.

Tracking takes one computation per chip length. Having a replica of the code, the receiver checks the match between the received chip, and the anticipated chip.

4.8 System Improvement

Inertial navigation system (INS) can be used in conjunction with magnetic PN signal to minimize possible errors [42, 43]. Since INS is relatively accurate in measuring displacement over short distances, it can be used to periodically check the accuracy of positioning by magnetic signal. It is specially useful in checking the accuracy within each run. The error of INS can be zeroed once the vehicle is sure that it is accurately reading the code. Every time a run of "0" or "1" ends, a receiver can assert that its reading at the end of the last run has been accurate. Combining the magnetic positioning system with INS will result in a very high accuracy positioning system.

Chapter 5

Conclusion

Automation of transportation systems is desirable because it is believed to not only reduce human errors and accidents but also reduce congestion and pollution. Specifically, researchers and engineers consider automation of major highways where automated vehicles can move safely, with smaller inter-vehicle spaces for efficient use of highways and vehicle fuel. The control structure for such automated system is divided between the automated road and the automated vehicles. The research presented here contributes to the automation of transportation systems by addressing the design and implementation issues of a coordination controller of automated vehicles in AHS.

We focused on the problem of coordination control for automated vehicles. That is, we considered the design and implementation issues of an automated vehicle controller for the purpose of coordinating the interaction of vehicles. Regarding issues of controller implementation, we proposed solutions for communication address resolution and vehicle positioning system. In the following, we summarize our work and discuss some extended

applications in each area of our research in order to provide possible directions and insights to future work in this field.

Vehicle Coordination Controller

We defined coordination safety for automated vehicles in terms of maintaining non-zero longitudinal and lateral distances at all times. We assumed that vehicles initially assume lateral and longitudinal margins that are determined with respect to their relative velocities and relative positions. Therefore, they would keep their distances as long as they are cruising at the same velocity. The vehicles should be concerned about coordination safety when they take actions other than *cruise*. Normally, a vehicle must coordinate its action with the actions of other vehicles such that safety margins are not violated but change according to the changes in relative velocity and positions of vehicles. We defined safety constraints for vehicle actions and designed a coordination controller that guarantees the safety constraints during vehicle actions. The coordination controller is an algorithm that is implemented through communication.

The proposed coordination controller can be used as a model for coordinating the interaction of unmanned aerial vehicles or an air traffic control system. For aerial vehicles and airplanes, the safety margins would be defined in three dimensions and a set of coordinated actions would need to be defined *a priori* for all vehicles such that when two aerial vehicles are about to violate the safety margins of one another, they would perform the designed coordinated actions and no collisions would occur. The aerial vehicles and airplanes must be able to communicate with one another and the communication range must

cover the safety margins. The communication structure of aerial vehicles and airplanes can take advantage of the address resolution schemes that we proposed for automated vehicles.

Communication

We considered the communication structure for vehicle networks and considered local area networks (LAN) and wide area networks (WAN) for intra- and inter-platoon communication, respectively. We focused on the WAN communication issues. Specifically, we considered the problem of address resolution, i.e., how a vehicle addresses another vehicle at a specific relative position as its communication party. We proposed two address resolution schemes for one-lane and multi-lane AHS. In a one-lane AHS, we noted that the relative position of vehicles do not change despite the fact that their velocities and distances change. Thus, we proposed using the order of vehicles in one lane to assign communication ID numbers to vehicles such that the numbers reflect the order of vehicles on the road. We also devised communication protocols that allow vehicles to maintain address information despite the vehicle actions of *join* and *split*. This scheme provides a model for (i) organizing a communication network through an external reference point, and (ii) maintaining the organized network by the network itself. That is, once the network is organized, it could be in charge of maintaining itself through communication protocols. Our proposed address resolution scheme provides a model for ad-hoc network organization and maintenance and the related issue of address resolution.

In a multi-lane AHS, we proposed a "flooding" scheme through a new medium access control scheme that takes advantage of the AHS road infrastructure capabilities.

Space-time division multi access (STDMA) is an innovative scheme to control the medium access based on the space location of the user. Every unit of space can contain at most one user at a time where the space is assigned a time division. Vehicles are assumed to be synchronized by the road infrastructure. Every vehicle transmits its communication address at the time division that is assigned to its unique location. This flooding repeats periodically to provide real-time address resolution to automated vehicles.

We considered mapping the user space to a unique time division because the vehicles can be synchronized by the road. However, the user space can be mapped to a unique frequency division, or a code division. In other words, the proposed STDMA scheme can be generalized to the *space division multiple access* (SDMA) scheme that is compatible with any multiple access scheme such as frequency division or code division. In the specific case where the multiple access scheme is time division, it would be STDMA.

In general, the geographical area where the users are located can be divided into smaller space divisions. The key element of the design is a one-to-one map between the space divisions and the bandwidth divisions of any multiple access scheme such as TDMA, CDMA, FDMA, etc. The system requirement is the knowledge of (i) position and (ii) the one-to-one map between the space divisions and the bandwidth divisions for medium access control (TDMA, FDMA, CDMA, etc.). STDMA uses SDMA with time divisions for medium access control to allow vehicles access the channel and broadcast their addresses. In other words, STDMA provides a means of "flooding" for address resolution in a multi-lane AHS.

The SDMA scheme allows the users to use their positioning information to access

the communication medium without contention or collision. Thus, SDMA can be used to implement a medium access control scheme in an ad-hoc network without using a controller entity or infrastructure. SDMA also allows a user to implicitly learn the position of other users.

SDMA can be used within different communication structures. It can be used for initializing and maintaining communication networks, i.e., for address resolution and channel allocation. Alternatively, SDMA can be used for data communication when users have homogeneous amount of data to transmit. A possible variation of the SDMA is to use the positioning information to assign "cluster heads" or "gates" for network management and routing purposes. The positioning information can be used for establishing a communication structure such as a communication hierarchy, where specific positions map to specific function nodes in the communication network.

In summary, SDMA provides a robust and reliable medium access control scheme for bootstrap organization of ad-hoc networks, given that every user has portioning information. We do not discuss the details of bandwidth efficiency because it highly depends on the specific application and communication structure design.

Magnetic Positioning System

Our address resolution scheme for multi-lane AHS assumes that every vehicle knows its position. This assumption was justified by our proposed positioning scheme for automated vehicles in AHS which uses pseudo-noise magnetic signals. The magnetic markers are at the center of every automated lane. We proposed that the markers carry a pseudo-

noise signal similar to the GPS signal. Reading the magnetic signal would allow accurate positioning of a vehicle in real-time. The advantage of positioning by magnetic markers is that it is more reliable than any currently available positioning system and its error bounds are subject to design. While the code structure and range measurement technique is very similar to GPS, the accuracy of this technique is remarkable. A novel feature of our proposed scheme is its error detection and correction capabilities. In conclusion, we compare positioning by PN coded magnetic markers with GPS.

Both GPS and magnetic positioning systems use PN codes with excellent correlation and autocorrelation properties. GPS uses digital radio waves, and magnetic positioning system uses magnetic pulses. The technology of receiving PN codes, detecting the phase, and mapping it to a physical position is very similar in both receivers. A GPS receiver integrates the entire hardware and software technology, but it depends on differential stations for high accuracy. The components of the magnetic positioning system, the magnetometer and the software, may not be physically integrated, but make an independent system.

Error probability distribution can be tabulated for magnetic positioning while the main source of error for GPS, multi-path, cannot be characterized. Unlike GPS, magnetic positioning system has the ability to detect errors and correct them. The accuracy of positioning by magnetic PN codes is subject to design and can be better controlled than the accuracy of GPS. Moreover, the positioning accuracy could vary over the road by varying the magnetic spacing. Finally, GPS receiver has a higher receiver cost.

The fact that our research project of *coordinating automated vehicles via communication* has involved different areas of technology in vehicle automation and communication

and positioning is a reflection of the rapid and interconnected growth of different technologies. However, each area of our research has its own potential for future work and extended applications.

Bibliography

[1] P. Varaiya, "Smart cars on smart roads: Problems of control," *IEEE Transactions on Automatic Control*, vol. 38, Feb 1993.

[2] D. N. Godbole, J. Lygeros, and S. Sastry, *Hierarchical Hybrid Control: a Case Study.* California PATH Research Report, UCB-ITS-PRR-95-9, 1995.

[3] K. Hedrick, J.Gerdes, V. Garg, and D. Maciuca, *Logitudinal Control Development for IVHS Fully Automated and Semi-Automated Systems: Phase III.* California PATH Research Report, UCB-ITS-PRR-97-20, 1997.

[4] "Intelligent vehicle highway systems," *Special Issue on Vehicle System Dynamics*, vol. 26, no. 4, 1996.

[5] P. Varaiya, *Models, simulation, and performance of fully automated highways.* California PATH Research Report, UCB-ITS-PRR-94-21, 1994.

[6] P. Varaiya and S. Shladover, "Sketch of an ivhs systems architecture," *Proceedings of the Vehicle Navigation and Information Systems Conference, (Dearborn, MI)*, pp. 909–922, October 20-23 1991.

[7] J. Lygeros, D. Godbole, and S. Sastry, "Verified hybrid controllers for automated vehicles," *IEEE Transactions on Automatic Control*, vol. 43, pp. 522–539, April 1998.

[8] D. N. Godbole, *Hierarchical Hybrid Control of Automated Highway Systems*. California PATH Research Report, UCB-ITS-PRR-95-8, 1995.

[9] F. Eskafi, *Hierarchical Hybrid Control of Automated Highway Systems*. California PATH Research Report, UCB-ITS-PRR-95-8, 1995.

[10] A. Hsu, F. Eskafi, S. Sachs, and P. Varaiya, *Protocol Design for an Automated Highway System*. Kluwer Academic Publishers, Boston, 1993.

[11] P. Li, R. Horowitz, L. Alvarez, J. Frankel, and M. Robertson, "An ahs link layer controller for traffic flow stabilization," *Transportation Research C*, vol. 5, no. 1, pp. 11–37, 1997.

[12] J. Frankel, L. Alvarez, R. Horowitz, and P. Li, "Safety-oriented maneuvers for ivhs," *Vehicles Systems Dynamics*, vol. 26, no. 4, pp. 271–299, 1996.

[13] L. Alvarez and R. Horowitz, *Traffic Flow Control in Automated Highway Systems*. California PATH Research Report, UCB-ITS-PRR-97-47, 1997.

[14] M. Broucke and P. Varaiya, "A theory of traffic flow in automated highway systems," *Transpn Research C*, vol. 4, pp. 181–210, 1996.

[15] R. Hall, "Longitudinal and lateral throughput on an idealized highway," *Transportation Science*, vol. 29, pp. 3499–3505, May 1995.

[16] J. Lygeros, *Hierarchical Hybrid Control of Large Scale Systems*. University of California at Berkeley, Ph.D. Thesis, 1996.

[17] P. Varaiya and R. Horowitz, "More on ivhs," *Vehicles Systems Dynamics*, vol. 26, no. 4, pp. 271–299, 1996.

[18] J. Lygeros, D. Godbole, and S. Sastry, "Multiagent hybrid system design using game theory and optimal control," *IEEE Conf. Decision and Control*, pp. 1190–1195, 1996.

[19] S. Sachs and P. Varaiya, "A communication system for the control of automated vehicles," September 1993.

[20] B. Foreman and P. Varaiya, "An infrared inter-vehicle communication system," 1995.

[21] S. Mahal, *Effects of Communication Delays on String Stability in an AHS Environment*. University of California at Berkeley, M.S. Thesis, 2000.

[22] T. Imielnski and H. F. Korth, *Mobile Computing*. Kluwer Academic Publishers, 1996.

[23] C. E. Perkins, *Mobile IP Design Principles and Practices*. Addison-Wesley Wireless Communications Series, 1998.

[24] J. Scourias. Available on WWW, URL http://ccnga.uwaterloo.ca/ jscouria/GSM/gsmreport.html, November 2000.

[25] A. Puri and P. Varaiya, "Simple results on communication with neighbors," 1992.

[26] *Title 21*. California Code.

[27] G. J. Holzmann, *Design and Validation of Computer Protocols*. Prentice Hall Software Series, 1991.

[28] G. J. Holzmann, *Basic Spin Manual*. Bell Laboratories, Murray Hill, NJ, Available on WWW, URL http://cm.bell-labs.com/cm/cs/what/spin/Man/Manual.html.

[29] S. Bana and P. Varaiya, "Address resolution in a one lane automated highway," *California PATH Research Report, UCB-ITS-PRR-00-21*, 1999.

[30] Y. Chen, *Effect of Communication Delays on the Performance of Vehicle Platoons*. University of California at Berkeley, Ph.D. Thesis, 1995.

[31] H.-S. Tan, R. Rajamani, and W.-B. Zhang, "Demonstration of an automated highway platoon system," *American Control Conference (ACC)*, pp. 1823–1827, 1998.

[32] M. K. Simon and J. K. Omura, *Spread Spectrum Communications Handbook, Revised Edition*. McGraw Hill, Inc., 1994.

[33] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *GPS Theory and Practice*. Springer-Verlag, 1994.

[34] J. J. S. Jr., "GPS signal structure and performance characteristics," *Papers published in Navigation*, vol. 1, pp. 29–54, 1980.

[35] R. J. Miliken and C. J. Zoller, "Principle of operation of NAVSTAR and system characteristics," *Papers published in Navigation*, vol. 1, pp. 3–20, 1980.

[36] E. G. Blackwell, "Overview of differential GPS methods," *Papers published in Navigation*, vol. 3, pp. 89–100, 1980.

[37] R. B. Langley, "Why is GPS signal so complicated?," *GPS World*, May/June 1990.

[38] R. B. Langley, "The federal radio-navigation plan," *GPS World*, March 1992.

[39] A. Kluesberg, "Precise differential positioning and surveying," *GPS World*, July/August 1992.

[40] R. B. Langley and A. Kluesberg, "The limitations of GPS," *GPS World*, March/April 1990.

[41] R. C. Dixon, *Spread Spectrum Systems with Commercial Applications, Third Edition.* Wiley Interscience, 1994.

[42] K. R. Britting, *Inertial Navigation Systems Analysis.* New York Wiley-Interscience, 1971.

[43] A. B. Chatfield, *Fundamentals of high accuracy inertial navigation.* New York Wiley-Interscience, 1997.