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

Hierarchical Hybrid Control: A Case Study

Datta N. Godbole
John Lygeros
Shankar Sastry

California PATH Research Report
UCB-ITS-PRR-95-9

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.

April 1995

ISSN 1055-1425

Hierarchical Hybrid Control: a Case Study *

Datta N. Godbole John Lygeros
Shankar Sastry

Intelligent Machines and Robotics Laboratory
University of California, Berkeley
Berkeley, CA 94720
godbole, lygeros, sastry@robotics.eecs.berkeley.edu

Abstract

A case study of the difficulties associated with the design of *hybrid* control systems is presented. We use the Intelligent Vehicle Highway System (IVHS) architecture of [1, 2], a system that involves both continuous state and discrete event controllers as our example of a hierarchical hybrid system. We point out that even though conventional analysis tools suggest that the proposed design should fulfill certain performance requirements simulation results show that it does not. We consider this as an indication that the conventional tools currently in use for the design and verification of control systems may be inadequate for the design of hierarchical control of hybrid systems.

1 Introduction

The term hybrid system is used to describe a large and varied class of systems. A large class of hybrid systems can be described by the architecture of Figure 1. A typical hybrid system is arranged in two (or more) layers [3, 4]. Different levels of abstractions of the plant model are used at each layer of the hierarchy. In the bottom layer the plant model is usually described by means of differential and/or difference equations. This level contains the actual plant and any conventional controllers working at the same level of abstraction. In the top layer the plant description is more abstract. Typical choices of description language at this level are finite state machines, fuzzy logic, Petri nets etc. Typically the controllers designed at this level are discrete event supervisory controllers (see e.g. [5]). The two levels communicate by means of an interface that plays the role of a translator between signals and symbols. As the techniques for control design and verification are well developed for the continuous and discrete systems, the design of the interface is of the utmost importance because it determines the way in which the combined system behaves.

*Research supported by the PATH program, Institute of Transportation Studies, University of California, Berkeley, under MOU-100

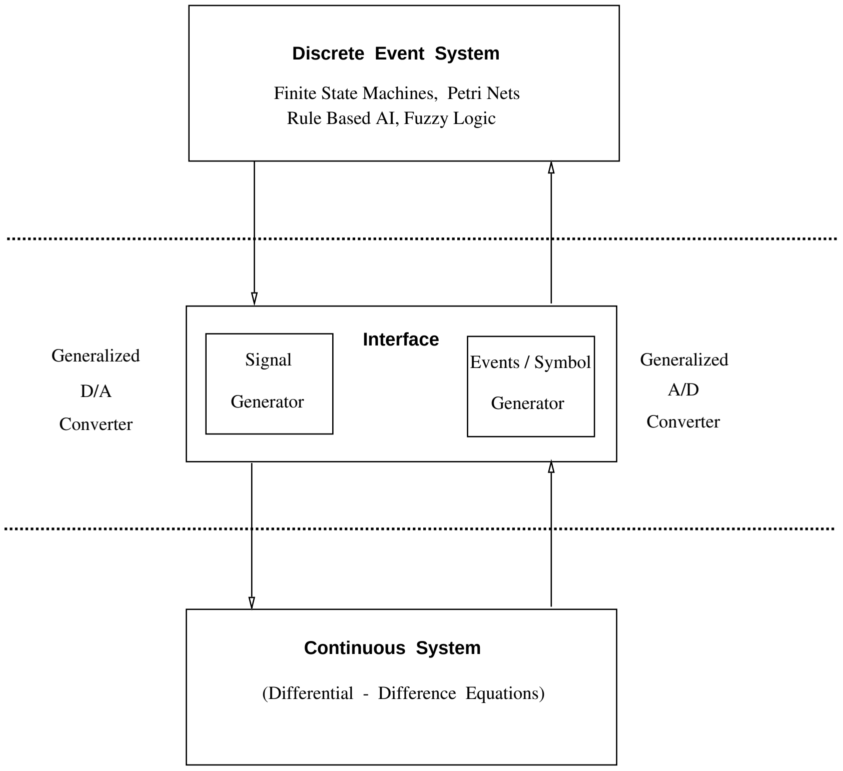
Discrete Event System
Finite State Machines, Petri Nets
Rule Based AI, Fuzzy Logic
Interface
Generalized D/A Converter
Signal Generator
Events / Symbol Generator
Generalized A/D Converter
Continuous System
(Differential - Difference Equations)

Figure 1: Hybrid System Architecture

Of course in a general hierarchical structure more than two levels may exist. If this is the case the structure of Figure 1 can be viewed as the interaction between any two of the layers. Alternatively the Discrete Event layer can be assumed to be a lumped version of all the layers that are not part of the continuous domain. Typically in a multilayered hierarchy as we move up the hierarchy system description gets more abstract (e.g. closer to linguistic), information is condensed (i.e. a signal in the higher levels codes many facts about the lower levels) and commands become more descriptive (i.e. a single command at a high level induces many actions at the lower levels).

The control architecture described above appears in wide variety of applications and forms the heart of most hybrid system formalisms. Switching controllers [6, 7], Intelligent Control [8, 9], Expert Control [10], Motion Control [11], among others, make use of the structure of Figure 1. For most of these systems the design approach has been "divide and conquer", that is the continuous and discrete controllers are designed independently and then combined by an interface which is designed with the specific problem. This is not a rule however as the literature also contains examples of systems that show active (on-line) design of the hybrid controllers (e.g. [12] and [13]) as well as attempts of formulating a consistent interface that is not case specific (e.g. [14]). In addition to theoretical formalisms a lot of work has also been done on techniques for simulating hierarchical, hybrid systems (e.g. [15] and [16]).

In this paper we present a case study in the design of hybrid control systems. We illustrate the problems associated with their design and verification by means of an example, the Intelligent Vehicle Highway System (IVHS) designed in the framework of the PATH project. Our goal is to illustrate that the "divide and conquer" approach to design is not always effective. The results we present indicate that for multilayered control structures there is a need for an independent proof of the combined system in addition to the usual proofs in the discrete and continuous domain. In the next Section we will outline the overall structure of the IVHS architecture that we are using in this study. More details on the parts of the design that are relevant to this study will be given in Section 3.

2 Intelligent Vehicle Highway System Architecture

One example where the hybrid system structure of Figure 1 can be found is the Intelligent Vehicle Highway System described in [1, 2]. The goal is to design a system that can significantly increase safety and highway capacity without having to build new roads, by adding intelligence to both the vehicles and the roadside. In order to achieve this the notion of "platooning" is introduced in [2]. It is assumed that traffic on the highway is organized in groups of tightly spaced vehicles, which are given the name platoons. Intuition suggests that doing this should lead to an increase in the capacity and throughput of the highway; indeed theoretical studies indicate that the capacity increase if such a scheme is implemented successfully will be substantial (as high as four times the current capacity). What may be more surprising is that this will be done without a negative impact on passenger safety. By having the vehicles within a platoon follow each other with a small intra-platoon separation of about 1 meter, we guarantee that if there is a failure and an impact is unavoidable, the relative speed of the vehicles involved in the collision will be small, hence the damage will be minimized. The inter-platoon separation, on the other hand, is large (of the order of 30 meters) to physically isolate the platoons from each other. The idea behind this is that, if needed, the platoons will have enough time to come to a stop before they collide. In addition a large separation guarantees that transient decelerations will be attenuated as they propagate down the freeway.

Clearly implementation of such a scheme would require the vehicles to be automatically controlled, as human drivers are not fast and reliable enough to be able to form platoons. The design of such a large scale control system pauses a formidable problem. In the architecture outlined in [2] the system is organized in five layers. The top layer, called the **network layer**, is responsible for the flow of traffic on the entire highway system[1]. Its task is to prevent congestion and maximize throughput by dynamic routing of traffic.

The block diagram of Figure 2 shows the remaining four layers of the hierarchy. The second layer, called the **link layer**, coordinates the operation of whole sections (links) of the highway. Its primary concern is to maximize throughput while maintaining safe conditions of operation. With these criteria in mind, it calculates an optimum platoon size and an optimum velocity for each highway section. It also decides which lanes the vehicles should follow to get to their destination as fast as possible. Finally, it monitors incidents on the highway and diverts traffic in order to minimize the impact of the incident on traffic flow and safety. Because the link layer bases its control actions on large numbers of vehicles, it inevitably has to use some form of aggregate information. Therefore it treats the vehicles in a section statistically rather than by considering the state of individual vehicles or platoons. Likewise,

[1]The highway system might consist of interconnection of several highways around an urban metropolis.

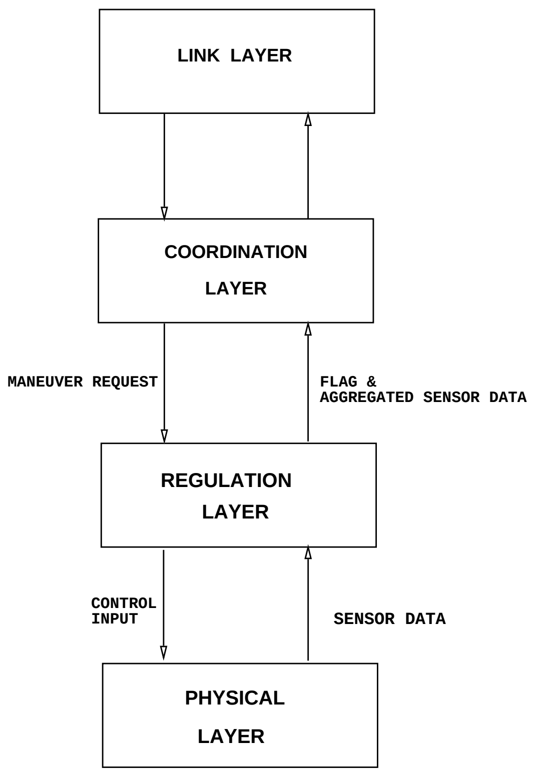

Figure 2: IVHS Architecture

the commands it issues are not addressed to individual vehicles but rather to all the vehicles in the section as a whole; a typical command would be "30% of the vehicles who wish to get off the highway at the next exit should change lane now" or "all platoons in this section should try to be 10 vehicles long".

The next level of hierarchy below the link layer is the **coordination layer**. It's task is to coordinate the operation of platoons with their neighbors. It receives the link layer commands and translates them to specific maneuvers that the platoons need to carry out. For example, it will ask two platoons to merge to a single platoon whose size is closer to the optimum or, given a command like "30% of the vehicles going to the next exit change lane now", it will decide which vehicles will comprise this 30% and split the platoons accordingly in order to let them out. The current design [17] uses protocols, in the form of finite state machines, to organize the maneuvers in a systematic way. They receive the commands of the link layer and aggregated sensor information from the individual vehicles (of the form "there is a vehicle in the adjacent lane"). They then use this information to decide on a control policy and issue commands to the regulation layer. The commands are typically of the form "accelerate to merge to the preceding platoon" or "decelerate so that another vehicle may move into your lane ahead of you".

Below the coordination layer in the control hierarchy lies the **regulation layer**. It's task is to receive the coordination layer commands and translate them to throttle, steering and breaking input for the actuators on the vehicle. For this purpose it utilizes a number of continuous time feedback control laws ([18, 19, 20, 21]) that use the readings provided by the sensors to calculate the actuator inputs required for a particular maneuver. The regulation layer occasionally needs to communicate with the coordination layer to inform it of the outcome of the maneuver.

The bottom layer is not part of the control hierarchy. It is called the **physical layer** and it contains the actual plant (in this case the vehicles with their sensors, actuators and communication equipment and the highway topology). For the purposes of simulation it can be assumed that the physical layer contains models of the actual physical quantities. From a hybrid systems point of view, the physical layer is merged with the regulation layer.

2.1 Hybrid control problem

The current paper focuses on the coordination and regulation layers and their interaction. Our aim is to demonstrate that the behavior of the overall hybrid control system may display characteristics, possibly undesirable, that are not predicted by analyzing the individual layers. In order to do this we will first describe briefly a possible IVHS design for which a dedicated simulation tool has been developed. In Section 3.1 the results in [21] that describe a possible design for the regulation layer will be presented while in Section 3.2 we will give an outline of the results in [17] where the design of the coordination layer is described. Finally in Section 3.3 we will discuss the results in [22] where an interface between these two layers is presented. References to alternative designs are also provided.

The references provide proofs of performance bounds for the individual layers. Still however no proof of performance for the overall design exists. The only means available for testing it at the moment is the SmartPath simulator [23]. In Section 4 of this report we will present the results of extensive simulations performed using this tool. It turns out that the performance of the combined system is not quite the expected. In particular, situations arise indicating limitation of the design that were not predicted by the individual layer analysis. Even though ways of designing around these shortcoming exist in most cases, it becomes apparent that if any faith is to be placed in the overall design some way of systematically determining its performance and limitations must be found. Unfortunately the failure of the standard tools (both in the finite state and the continuous domain) to predict the limitations discovered by means of simulation indicates that new, more powerful tools are needed to achieve this goal. Our current work focuses on the development of such tools.

3 Design of Individual Layers

In this Section we will give a brief description of the multilayered control design that was implemented in the SmartPath simulator. We will focus our attention on the regulation (continuous) and coordination (discrete) layers and their interface.

It should be noted that extensive work has also been done in the link layer design (see [24]) while work is already in progress for the network layer. The link layer control scheme has also been added to the SmartPath simulator. However, the simulation results presented here were obtained using a simplified version of SmartPath where the detailed link layer design is substituted by a small number of simple abstractions; for example such an abstraction might be “all vehicles coming in the freeway should move to the fast lane and stay there until they are one mile from their exit”. The justification for this simplification is that we are interested in the interaction of the two bottom layers of the control architecture and the problems it pauses as a hybrid system and do not want the link layer intelligence to “pollute” their behavior. A few more thoughts on the merits of a simplistic abstraction for the link layer are given in the concluding Section.

It should also be noted that the IVHS design presented here is by no means unique or optimal, but is sufficient to illustrate our point. References to possible alternative designs will be given throughout the paper.

3.1 Continuous Layer

Conventional differential equation models are used to design continuous time control laws at this level, which contains both the regulation and the physical layer of Figure 2. The advantage of this framework is that it supports well established verification techniques in the form of mathematical proofs. The procedure used for the design of most of the control laws that appear in the IVHS architecture in question is to approximate the dynamics of a vehicle by a linear or feedback linearizable system for which controllers are designed that are robust enough to take care of any unmodeled dynamics. This process simplifies the task of analyzing the stability and performance of the algorithms. The regulation layer contains six such control laws:

- Leader Control law:
  We assume that the platoon leaders will implement an Autonomous Intelligent Cruise Control[2] (AICC) law like the one presented in [21]. The goal of this controller is to maintain safe inter-platoon spacing (which in [21] is taken as a constant time headway of 1 second) and track the optimal velocity determined by the link layer (typically 60-65 miles per hour) if possible. Because the objectives of this controller are very similar to the objectives of human drivers, this control law is used as the default for a leader, i.e. it is the law implemented unless there is a specific command to do otherwise.

- Follower control law:
  This is the default control law for the followers in a platoon (see [18, 20]). Its task is simply to track the velocity of the vehicle in front of it in the same platoon, while keeping a close distance behind it (1 meter in this case).

- Merge control law:
  Merge is the action taken by two platoons that want to become one. The following platoon, that requests the merge, accelerates to catch up with the leading platoon and join it. This acceleration is done according to a reference trajectory which is calculated based on the assumption that the preceding platoon will be moving at constant velocity throughout the entire maneuver. State feedback is then used to stabilize the closed loop system about this trajectory and hence eliminate the effect of any acceleration or deceleration of the leading platoon (see [21]).

[2]See [25] for a different design of AICC control law.

- Split Control law:
  Split is exactly the opposite of merge. A follower becomes the leader of all the cars that follow it (in the same platoon) and decelerates until it reaches safe inter-platoon distance from the mother platoon. Like the merge control law, a reference open loop trajectory is calculated and then state feedback is used to guarantee asymptotic tracking.

- Change lane control law:
  Apart from the obvious lateral (steering) action, change lane also requires longitudinal action in terms of aligning the vehicle that wishes to change lane with a proper gap in the target lane. This action is in principle similar to a split as discussed in [21]. Both the longitudinal and lateral components of this maneuver are carried out by controllers that are designed in the same way as the merge or split controllers.

- Lateral control law:
  The objective of this law is to keep the vehicle in the center of the lane. This is done by means of a frequency shaped linear quadratic (FSLQ) regulator ([19]).

It can be shown that individually all the control laws described above lead to a closed loop system that is stable and capable of performing the desired task adequately. The proofs and simulation results of this claim can be found in the references. In addition, some of these controllers have been tested experimentally on actual vehicles with satisfactory results.

The continuous layer in the sense of Figure 1 also contains the physical layer of Figure 2. The work presented here was carried out under the assumption that the operation of the sensors and actuators of the physical layer of all vehicles is perfect, i.e. there are no time delays, no steady state tracking error, no false measurements, etc. The only restrictive assumption we make is that we impose limits on the ranges of the sensors and actuators. These limits are based on experimental data and assume perfect external conditions (dry road, and perfect weather). Therefore they are, if anything, optimistic given the current technology. They are summarized in table 3.1.

Table 1: Constraints on Actuators and Sensors

Max. Acceleration	3	m/s^2
Max. Deceleration	-5	m/s^2
Distance Sensor Front	60	m
Distance Sensor Rear	30	m
Distance sensor Adjacent Lane, Front	30	m
Distance sensor Adjacent Lane, Rear	30	m

As will become apparent in Section 4 these limits play a crucial role in the behavior of the combined system. In particular, we can identify a region in the state space of the lead vehicle, corresponding to certain severe disturbances, from which the lead control can not recover. (see [21] for details.)

3.2 Discrete Layer

The discrete layer works at a more abstract level than the continuous layer. Many different formalisms exist for describing this layer; standard choices include finite state machines, Petri or Neural Nets and Fuzzy Logic among others. The framework of most of these techniques supports methods of carrying out proofs of correctness. For example, a very effective proof method is the one used for the verification of finite state machines. It is based on the fact that, because of the finiteness of the state, it is possible to enumerate all possible traces that the machine accepts and hence verify that all possible sequences of events possess certain desirable properties. Tools exist in the form of computer programs that perform this verification task automatically. Of course the actual tools are a lot more sophisticated than the simplistic description given above and are designed to be computationally efficient, a very useful property as the machines in question often have hundreds of thousands or even millions of states.

In our example we will assume that the discrete layer contains only the coordination layer. As mentioned in the introduction a simple abstraction will be used for the link layer. The task of this discrete layer is to provide a consistent way of coordinating the maneuvers of adjacent platoons so that they are effective (vehicles get to their destination) and efficient (capacity is maximized) without compromising safety. To facilitate the analysis and keep the problem tractable, the design in [17] distinguishes only three maneuvers: *Merge* to form a single platoon from two platoons, *Split* to do the opposite and *Change* to move a vehicle from one lane to the other. To further simplify the situation it is assumed that each platoon can only be involved in one maneuver at a time and that only free agents, that is one car platoons, can change lane. Clearly these assumptions lead to a rather restricted set of possible behaviors. It may be possible to obtain better designs by relaxing some of them (see for example [26]). However, the restrictions allow us to keep track of the problem and prove that the design possesses certain desirable properties.

The coordination layer controller is supposed to carry out these three maneuvers efficiently and safely. To accomplish this, the coordination layer of each vehicle exchanges messages with neighboring vehicles according to certain protocols (which in [17] are modeled by finite state machines). When mutual agreement (coordination) is reached, it commands the regulation layer to carry out the appropriate maneuver. The protocols were verified, to posses certain desirable properties using an automated software verification tool, COSPAN [27]. For the purpose of this verification, the behavior of the continuous layer was abstracted by a set of finite state machines, that are supposed to model the behavior of the sensors and the conventional controllers from the protocol point of view. So the verification proved that the protocol logic is “correct” when coupled with the abstraction of the continuous layer. How closely the abstractions match the behavior of the actual system is still an open question however.

3.3 Interface

Interface design is the most challenging part of a hybrid system since it straddles both the continuous and the discrete world. Good interface design is very important as it determines to a large extend what one can prove about the combined system. It may also help us extend advantages of the discrete event system theory (e.g. ease of computation) to the continuous domain and conversely (e.g. derive mathematical proofs in discrete space from equivalent proofs on continuous state spaces).

In applications such as ours, the interface is usually designed last. The disadvantage of this approach is that it leads to interfaces that are case specific and whose performance is limited by the limitations of the rest of the design. One of the goals of this research is to develop a technique for systematically designing interfaces with desirable properties for general systems.

In our example the interface is a finite state machine whose transitions depend upon the commands from the coordination layer, the readings of the sensors (physical layer responses) and the state of the continuous controllers. It plays a dual role. On the one side it acts as a symbol to signal translator and therefore directly influences the evolution of the continuous system. It receives the coordination layer commands (symbols) and uses them to switch between the different continuous layer controllers (signals). In addition it keeps track of which of these controllers needs to be initialized (symbol) and carries out this initialization by directly changing the controller state (signal). In the other direction the interface acts as a signal to symbol translator. It processes the sensory information (signal) and presents it to the coordination layer in an aggregate form compatible with the finite state machine formalism (symbol). It also monitors the evolution of the continuous system (signal) and decides if the maneuver in progress is safe or not. If at any stage the maneuver becomes hazardous it aborts it, notifies the coordination layer of its decision (symbol) and switches to a different continuous control law that will get the system back to a safe configuration.

Unfortunately there is no systematic way of verifying the complete interface. [22] describes the verification of the discrete part of the interface, where the continuous state considerations are ignored or abstracted. However, the addition of the actual continuous effects to the framework in a consistent way is not supported by the current theory.

The only way of testing the behavior of the combined system at the moment is by simulation. Even though successful simulation results are not nearly as good as a mathematical proof of the properties of the system, a failure in the simulation can be considered as a proof of a shortcoming of the actual design. This approach will be used in the next section to point out problems in the current IVHS design presented here and try to determine their causes.

4 Properties of Combined System

The design described above was put to test by implementing it in simulation. The result was a dedicated simulator, SmartPath, which is described in [23]. An effort was made to make the simulator as flexible and modular as possible so that changes in the design (e.g. alternative regulation layer controllers) can be implemented easily. Using this tool, long simulations for a variety of initial conditions and inputs and a few choices of simplified link layer design were carried out. Contrary to our expectation we observed quite a few scenarios where the performance of the system was inferior to the one predicted by the individual level analysis, for example unpredicted car crashes occurred. We now document these different scenarios.

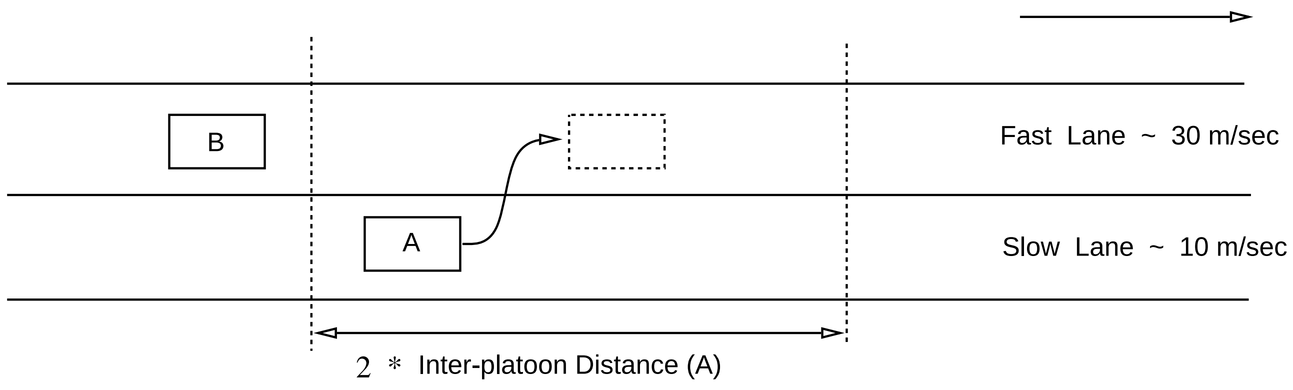

Figure 3: Change lane from a slow to a fast lane

4.1 Changing from a slow to a fast lane

According to the coordination layer design, only free agents (single vehicle platoons) are allowed to change lane. Before a vehicle initiates a lane change it looks (through its sensors) to the adjacent lane to make sure that there is room for it there. If no vehicle is visible in the sensor range the move is initiated immediately. If a vehicle is found and its distance is less than the safe inter-platoon spacing communication is established to coordinate the maneuver. This goes on until a gap twice as large as the safe inter-platoon spacing is found. Then the lane change takes place in the middle of this gap.

In most situations this arrangement should cause no problems. Indeed both the protocol that coordinates the maneuver and the regulation layer controllers that align the free agent with a gap in the next lane have been proven to perform well. Consider however the scenario shown in Figure 3. Free agent A switches from a slow lane to a fast lane. During the change a gap big enough for A to move into is present in the fast lane (for example no vehicle is visible in the sensor range). It is conceivable however that a vehicle (denoted by B) is present in the fast lane behind A, which, after the lane change is complete, finds itself just outside A's rear sensor range (say $35m$) and moving a lot faster than A (say $30m/s$ as opposed to $10m/s$). It turns out that the AICC lead controller is incapable of recovering from such drastic initial conditions, so a crash is inevitable.[3]

We were able to recreate this kind of crash in a relatively light traffic situation by asking many vehicles to change lane at the same time in order to leave at an intersection. As the vehicles need to be free agents before they change lane many splits were carried out, which inevitably lead to a large deceleration in the lane of origin. This deceleration was not present in the target lane however, so, after a few seconds, the vehicles in the target lane found themselves moving a lot faster than the vehicles in the origin lane. Crashes of the kind described above were then observed. An example of this scenario is given in Figures 7 (where each point represents the position of a vehicle at the given time instant).

[3]Note that the safe inter-platoon spacing for a vehicle is affine in its velocity. In figure 3, the safe inter-platoon spacing is 20 m for vehicle A but 40 m for vehicle B.

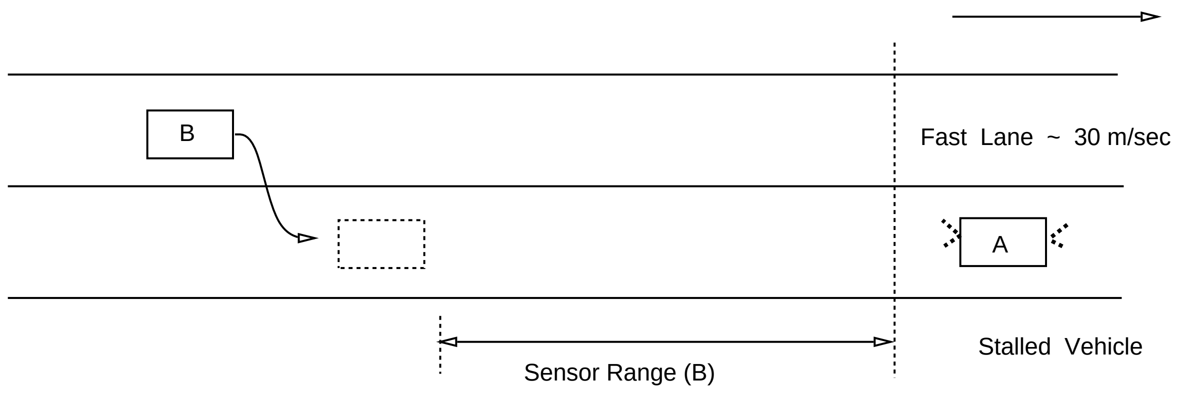

Figure 4: Change lane from a fast to a slow lane

4.2 Changing from a fast to a slow lane

This is a mirror image of the situation of previous case. In this case (Figure 4), the vehicle B changes lane and hits a slow moving (or stalled) vehicle A from behind. During the lane change vehicle A was just outside the range of the forward adjacent lane sensor of B. After the change was complete the front sensor of B reveals the presence of A but it is already too late to stop. The scenarios that might lead to such a crash are the same as the ones of Section 4.1, with the origin and target lanes interchanged.

It turns out that it is very difficult to recreate a crash like this using the current version of SmartPath. Figure 13 shows an extreme case where a stalled vehicle is seen at the edge of the front sensor range (60 m). The optimum velocity commanded by the link layer is too high for the second vehicle to stop in time and therefore a crash is unavoidable.

4.3 Merging to a decelerating platoon

With the merge control law described in Section 3.1, changes in the velocity of the front platoon should cause no problem as the state feedback should take care of any deviations from the desired trajectory. However, the actual trajectory *will* deviate from the desired one if the limits of the actuators (throttle and brake) are reached. To avoid this possibility, the interface aborts the maneuver when it detects the danger of actuator saturation (see [22]). After aborting the maneuver the system should find itself in a position from which it can continue safely under the AICC lead control law. The simulation indicates that under extreme conditions this may not be true and merging may cause a major hazard. We were able to recreate two such scenarios.

In the first case we created a large deceleration in a lane by making the simplified link layer ask many of the vehicles in the adjacent lanes to move to the lane in question. The vehicles already in the lane were then forced to decelerate to create space for the incoming vehicles. In the second scenario we caused a large deceleration by asking many of the vehicles in a densely occupied lane to exit. As already discussed this caused a number of splits in the platoons as the vehicles tried to become free agents in order to change lanes. In both situations the deceleration built up enough to cause saturation of the actuators. Therefore merging vehicles upstream of the disruption were forced to abort their maneuvers. This led to

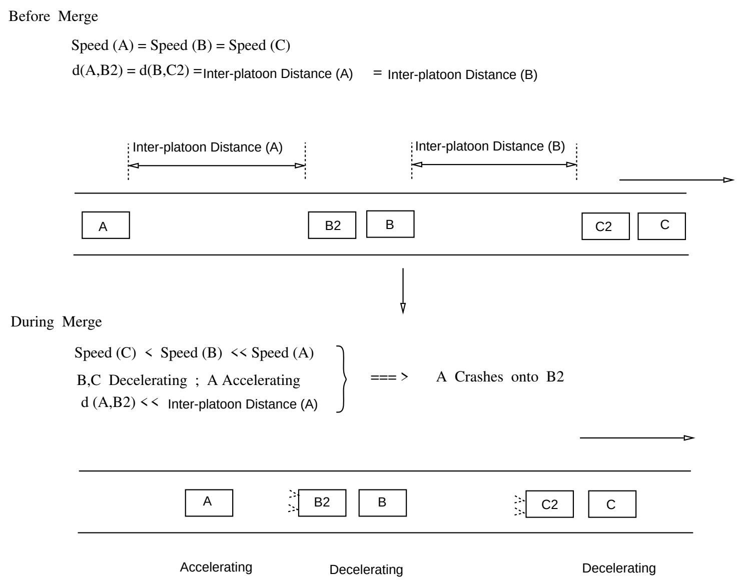
Before Merge
Speed (A) = Speed (B) = Speed (C)
d(A,B2) = d(B,C2) = Inter-platoon Distance (A) = Inter-platoon Distance (B)
Inter-platoon Distance (A)
Inter-platoon Distance (B)
A
B2
B
C2
C
During Merge
Speed (C) < Speed (B) << Speed (A)
B,C Decelerating ; A Accelerating
d (A,B2) << Inter-platoon Distance (A)
===> A Crashes onto B2
A
B2
B
C2
C
Accelerating
Decelerating
Decelerating

Figure 5: Crash during Merge maneuver

a situation where vehicles (like vehicle A in Figure 5) found themselves close to the preceding platoon (B-B2) which is already decelerating at the maximum rate, while moving faster than it. Hence, even though A decelerates at saturation level as well, a crash is unavoidable.

A situation where such a crash is observed is shown in Figures 9 and 10.

Note: Vehicle crashes associated with merge maneuver were also predicted by A. Hitchcock (See [28]) using *Fault Tree analysis* method.

4.4 Multiple lane changes from a single platoon

The situation described above, where many vehicles wish to change out of a lane thus creating multiple splits, caused yet another kind of crash. The scenario is outlined in Figure 6.

A platoon breaks up so that vehicles B, C and D can leave the lane. The current interface declares a split maneuver complete the moment the deceleration begins. This allows the coordination layer to initiate a lane change maneuver while the split deceleration is taking place. This is done to improve the rate at which vehicles leave the lane. However, it is conceivable that three splits will occur before a lane change is initiated. So B, C and D will all become free agents one after the other. This will cause no problem as long as the vehicles stay in the lane. Suppose though that all three of them want to move to the next lane. As discussed above they will all look at the target lane through their sensors, and, assuming they find it empty will all start moving over at roughly the same time. If D reaches the target lane first it will start accelerating to get back to the optimum speed. In the meantime C, which has been decelerating because of the split maneuver, reaches the lane fractions of a second later. As a result D finds itself very close to C and moving slightly faster. It is very likely that this situation might lead to a crash. Indeed Figure 11 shows the results of a simulation where such a crash was observed.

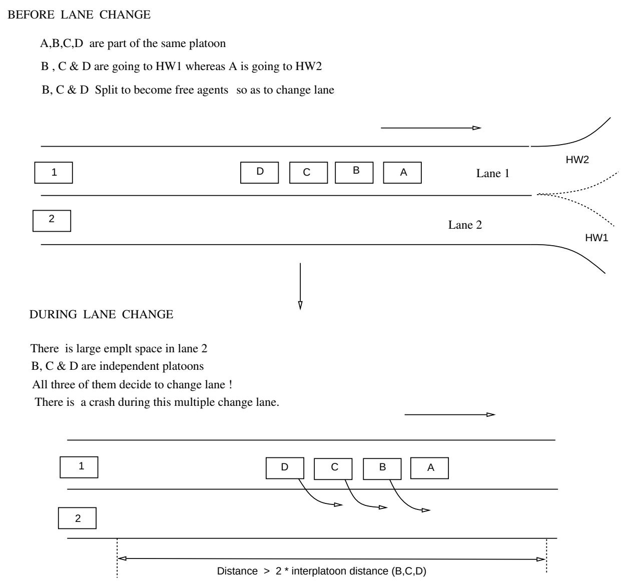
BEFORE LANE CHANGE
A,B,C,D are part of the same platoon
B , C & D are going to HW1 whereas A is going to HW2
B, C & D Split to become free agents so as to change lane
1
D
C
B
A
Lane 1
HW2
2
Lane 2
HW1
DURING LANE CHANGE
There is large emplt space in lane 2
B, C & D are independent platoons
All three of them decide to change lane !
There is a crash during this multiple change lane.
1
D
C
B
A
2
Distance > 2 * interplatoon distance (B,C,D)

Figure 6: Multiple Change Lane

4.5 Crash Analysis

The common feature of all the crashes described above is that they are caused by continuous layer performance not accounted for by the discrete layer. In all cases this leads to the discrete layer making requests that are incompatible with the current situation of the continuous layer, such as a potentially dangerous maneuver. The situations in Sections 4.1 and 4.2 arise because the limits on the sensor range and the continuous time controller performance are not represented adequately by the corresponding coordination layer abstractions. The same is true for the situation in 4.3, only instead of the sensor limits the problem here arises because of the actuator limits. Finally the problem in Section 4.4 is caused by the fact that the coordination layer implicitly assumes that during a lane change vehicles in the origin lane will already be at a safe distance, an assumption related to the continuous layer. As a result it does not establish communication with them and therefore is oblivious of their intend to change lanes.

Clearly most of these crashes could have been avoided by changes in the control design. For example to exclude the scenario in Section 4.4 we could ask the interface to declare the split maneuver complete (and allow the initiation of a new maneuver) *after* the splitting platoon has decelerated all the way to safe spacing. This fact however is besides the point. For one, ad-hoc changes like that might have unpredictable effect on other aspects of the system performance (e.g. capacity). More importantly the simulation results demonstrated that, even though the individual layers have been verified independently to exclude certain undesirable kind of behavior, the combined system is capable of producing them. It should also be noted here that the list of faults presented above is by no means exhaustive. The fact that we were able to identify only the above scenarios as dangerous does not mean that there do not exist other situations that may lead to unpredictable catastrophes. These observations naturally lead to the question to what extend can one trust the conventional discrete and continuous verification techniques when it comes to hybrid systems. Clearly if any faith is to be placed in the IVHS design presented here a proof of its performance claims is needed. The above indicates that such a proof is not possible using conventional tools.

As already mentioned the link layers used in the above simulation were extremely simple. One might feel that the crashes could have been avoided with a more realistic and intelligent link layer design. Even if this is true it still does not change the fact that the coordination-regulation layer interaction needs to be studied more, as the performance claims made by the verification of the individual layers are independent of the link layer design.

It should be noted here that the above discussion should not give the impression that car crashes are a common occurrence in the proposed IVHS design. In fact they are rather rare and occur only under special, extreme conditions, which we mostly set up specifically in order to observe the crashes. Statistically speaking the crash of the kind described in Section 4.3 are the most common. Crashes of the kind described in Section 4.1 are rather rare and only happen under extreme conditions. The same is true for the crashes described in Section 4.4 which are even more rare. Finally crashes like the ones in Section 4.2 are very rare indeed and are only observed under scenarios that were artificially created.

5 Concluding Remarks

Summarizing the above results, the regulation and coordination (continuous and discrete) layers of the IVHS architecture shown in Figure 2 were independently developed and were then combined by an interface. At each level (discrete, continuous and interface) verification of desirable properties was carried out using the appropriate tools. However, contrary to expectations, problems were detected in the combined system performance by simulation. The problems were due to the fact that the discrete layer can only comprehend abstractions of the continuous layer. As a result it can not take sufficiently into account certain continuous layer parameters, which in our case were the sensor and actuator ranges and the controller performance bounds. These observations led to the conclusion that in order to fully trust the design we need verification tools that test the performance of the combined *hybrid* system.

Computer aided verification of timed [29] and hybrid systems [30, 31] is currently an active area of research. For verification, the actual behaviors of the hybrid systems are usually abstracted into a finite set of behaviors. Thus one trajectory of the abstraction will include many possible system trajectories which are “close” to it in some sense. This is a promising development as for example it could be used in the context of IVHS to prove that a particular design will not produce any crashes in normal mode. It should be noted however that the application of such a technique to the current design may be inconclusive as the failure of the verification may be attributed to either a faulty design or over-abstraction.

In terms of progress in the IVHS part of the problem, the simulation results may also suggest considerations to be taken into account by the link layer. For example, while dealing with an accident downstream, it might seem natural to declare all vehicle in a platoon to be free agents in order to try and get as many of them out of this lane as possible. But this will create successive splits and excessive decelerations for vehicles further upstream. Under these conditions crashes like the ones described in Sections 4.1 and 4.3 will be possible. Similarly, downstream from an accident vehicles will want to move from the congested (free) lane to the nearly empty lane (where the accident was). This may again lead to the same type of crashes. Finally, the risks associated with the merge maneuver may be unavoidable without extensive redesign, as the cause of deceleration (e.g. splitting of platoons downstream etc) are uncontrollable and can occur any time during the merge [4]. These results also indicate that the performance of the current design, even with the link layer, may not be robust enough to deal with certain situations. We collectively refer to such extreme conditions as "degraded modes of operation". More work is currently in progress to provide alternatives to the current design in order to deal with these scenarios.

Acknowledgment: The authors would like to thank Farokh Eskafi, Delnaz Khorramabadi, Dr. Bobby Rao and Prof. Pravin Varaiya for helpful discussions providing insight into the problem.

References

[1] P. Varaiya and S. E. Shladover, "Sketch of an IVHS systems architecture," Tech. Rep. UCB-ITS-PRR-91-3, Institute of Transportation Studies, University of California, Berkeley, 1991.

[2] P. Varaiya, "Smart cars on smart roads: problems of control," *IEEE Transactions on Automatic Control*, vol. AC-38, no. 2, pp. 195–207, 1993.

[3] A. Nerode and W. Kohn, "Models for hybrid systems: Automata, topologies, controllability, observability," in *Hybrid System* (R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds.), pp. 317–356, New York: Springer Verlag, 1993.

[4] R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, *Hybrid Systems*. Springer Verlag, 1993.

[5] P. J. G. Ramadge and W. M. Wonham, "The control of discrete event dynamical systems," *Proceedings of IEEE*, vol. Vol.77, no. 1, pp. 81–98, 1989.

[6] A. S. Morse, "Supervisory control of family of linear set point controllers," in *IEEE Control and Decision Conference*, pp. 1055–1060, 1993.

[7] K. S. Narendra and J. Balakrishnan, "Improving transient response of adaptive control systems using multiple models and switching," in *IEEE Control and Decision Conference*, pp. 1067–1072, 1993.

[8] M. Lemmon, J. A. Stiver, and P. J. Antsaklis, "Event identification and intelligent hybrid control," in *Hybrid System* (R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds.), pp. 268–296, New York: Springer Verlag, 1993.

[9] K. M. Passino and P. J. Antsaklis, "Modeling and analysis of artificially intelligent planning systems," in *An Introduction to Intelligent and Autonomous Control* (P. J. Antsaklis and K. M. Passino, eds.), pp. 191–214, Boston: Kluwer Academic Publishing, 1993.

[4] There exists a different scheme of organizing traffic without merge maneuver due to A. Hitchcock [26]. This architecture is currently under investigation. It will probably still be vulnerable to crashes during lane changes

[10] K. M. Passino and A. D. Lunardhi, "Stability analysis of expert control systems," in *IEEE Control and Decision Conference*, pp. 765–770, 1993.

[11] R. W. Brockett, "Hybrid models for motion control systems," in *European Control Conference*, 1993.

[12] A. Nerode and W. Kohn, "Multiple agent hybrid control architecture," in *Hybrid System* (R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds.), pp. 297–316, New York: Springer Verlag, 1993.

[13] V. Borkar, M. S. Branicky, and S. K. Mitter, "A unified framework for hybrid control." (preprint).

[14] R. W. Brockett, "Pulse driven dynamical systems." preprint.

[15] A. Back, J. Guckenheimer, and M. Myers, "A dynamical simulation facility for hybrid systems," Tech. Rep. 92-6, Cornell University, April 1992.

[16] L. Tavernini, "Differential automata and their simulators," *Nonlinear Analysis, Theory, Methods and Applications*, vol. 11(6), pp. 665–683, 1987.

[17] A. Hsu, F. Eskafi, S. Sachs, and P. Varaiya, "The design of platoon maneuver protocols for IVHS," Tech. Rep. UCB-ITS-PRR-91-6, University of California, Berkeley, 1991.

[18] J. K. Hedrick, D.McMahon, V. Narendran, and D. Swaroop, "Longitudinal vehicle controller design for IVHS system," in *American Control Conference*, pp. 3107–3112, 1991.

[19] H. Peng and M. Tomizuka, "Vehicle lateral control for highway automation," in *American Control Conference*, pp. 788–794, 1990.

[20] S. Sheikholeslam and C. A. Desoer, "Longitudinal control of a platoon of vehicles," in *American Control Conference*, pp. 291–297, 1990.

[21] D. Godbole and J. Lygeros, "Longitudinal control of the lead car of a platoon," Tech. Rep. PATH Memorandum 93-7, Institute of Transportation Studies, University of California, Berkeley, 1993.

[22] J. Lygeros and D. Godbole, "An interface between continuous and discrete-event controllers for vehicle automation," Tech. Rep. PATH Memorandum 93-8, Institute of Transportation Studies, University of California, Berkeley, 1993.

[23] F. Eskafi, D. Khorramabadi, and P. Varaiya, "Smartpath: An automated highway system simulator," Tech. Rep. PATH Memorandum 92-3, Institute of Transportation Studies, University of California, Berkeley, 1992.

[24] B. Rao and P. Varaiya, "Roadside intelligence for flow control in an IVHS," tech. rep., PATH Technical Memo, Institute of Transportation Studies, University of California, Berkeley, 1993.

[25] P. Ioannou, Z. Xu, S. Eckert, D. Clemons, and T. Sieja, “Intelligent cruise control: Theory and experiments,” in *IEEE Control and Decision Conference*, pp. 1885–1890, 1993.

[26] A. Hitchcock, “A specification of an automated freeway with vehicle-born intelligence,” tech. rep., PATH Technical Memo, Institute of Transportation Studies, University of California, Berkeley, 1994.

[27] Z. Har’El and R. Kurshan, *Cospan User’s Guide*. AT&T Bell Laboratories, 1987.

[28] A. Hitchcock, “Casualties in accidents occuring during split and merge maneuvers,” tech. rep., PATH Technical Memo 93-9, Institute of Transportation Studies, University of California, Berkeley, 1993.

[29] R. Alur, C. Courcoubetis, and D. Dill, “Model checking for real-time systems,” *Logic in Computer Science*, pp. 414–425, 1990.

[30] R. Alur, C. Courcoubetis, T. A. Henzinger, and P. H. Ho, “Hybrid automaton: An algorithmic approach to the specification and verification of hybrid systems,” in *Hybrid System* (R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds.), pp. 209–229, New York: Springer Verlag, 1993.

[31] A. Puri and P. Varaiya, “Decidebility of hybrid systems with rectangular differential inclusions.” (preprint).

A Simulation Results

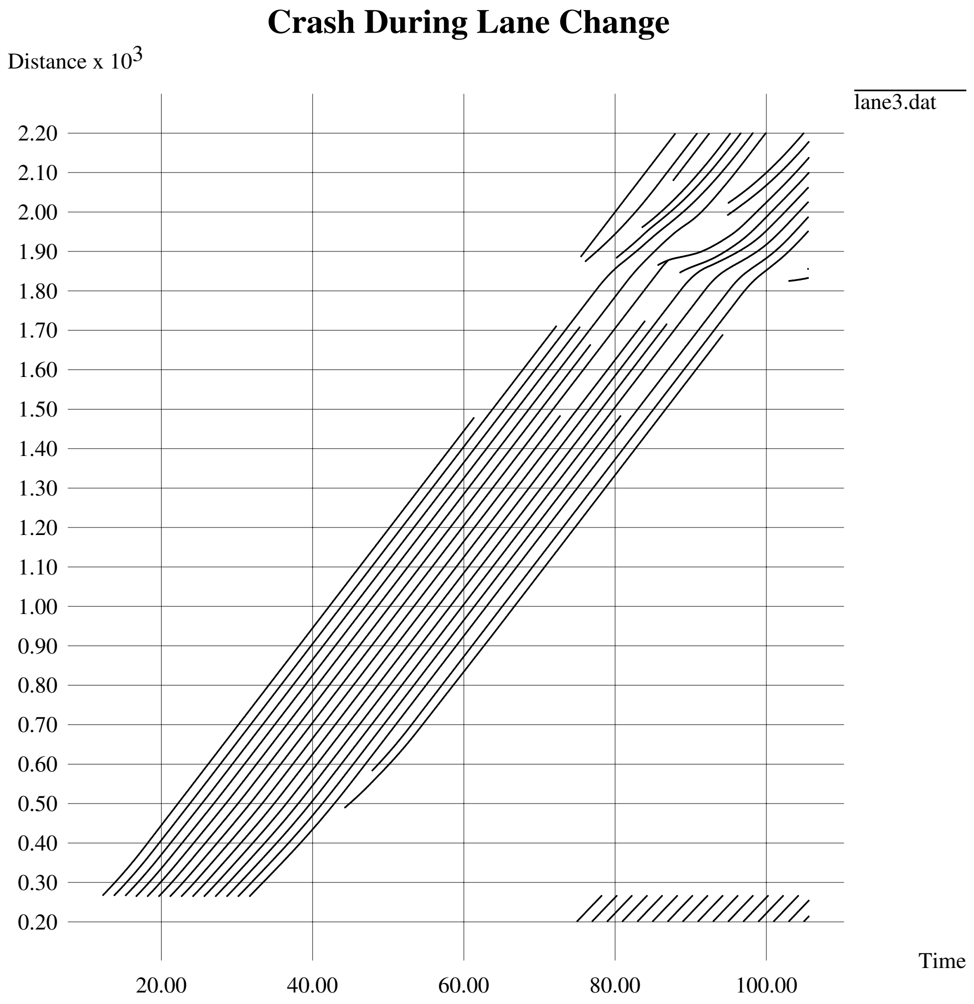

Figure 7: Lane change from slow to fast lane: the crash occurs at 86.91 seconds at a distance of 1870 meters. The next figure shows a close up view of the crash

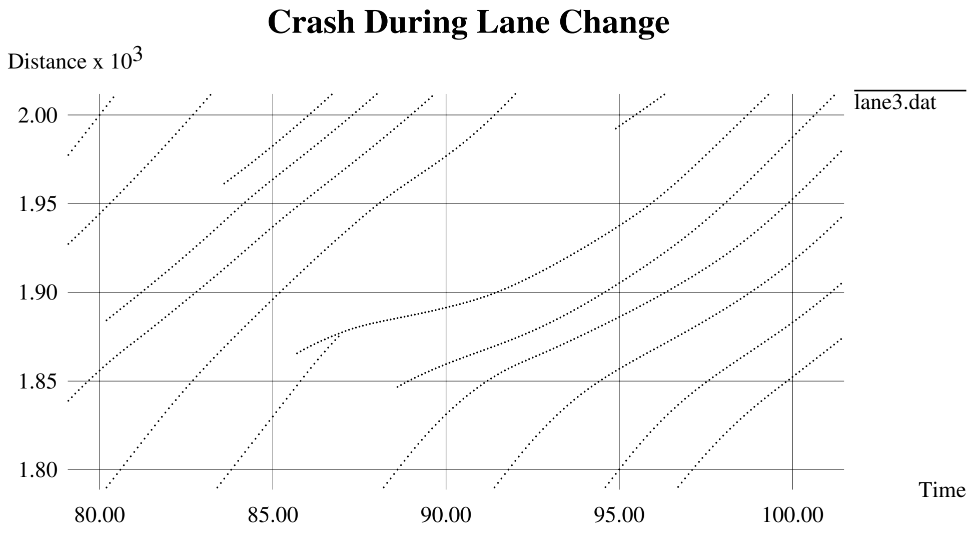

Figure 8: Lane change from slow to fast lane: close up view of the crash

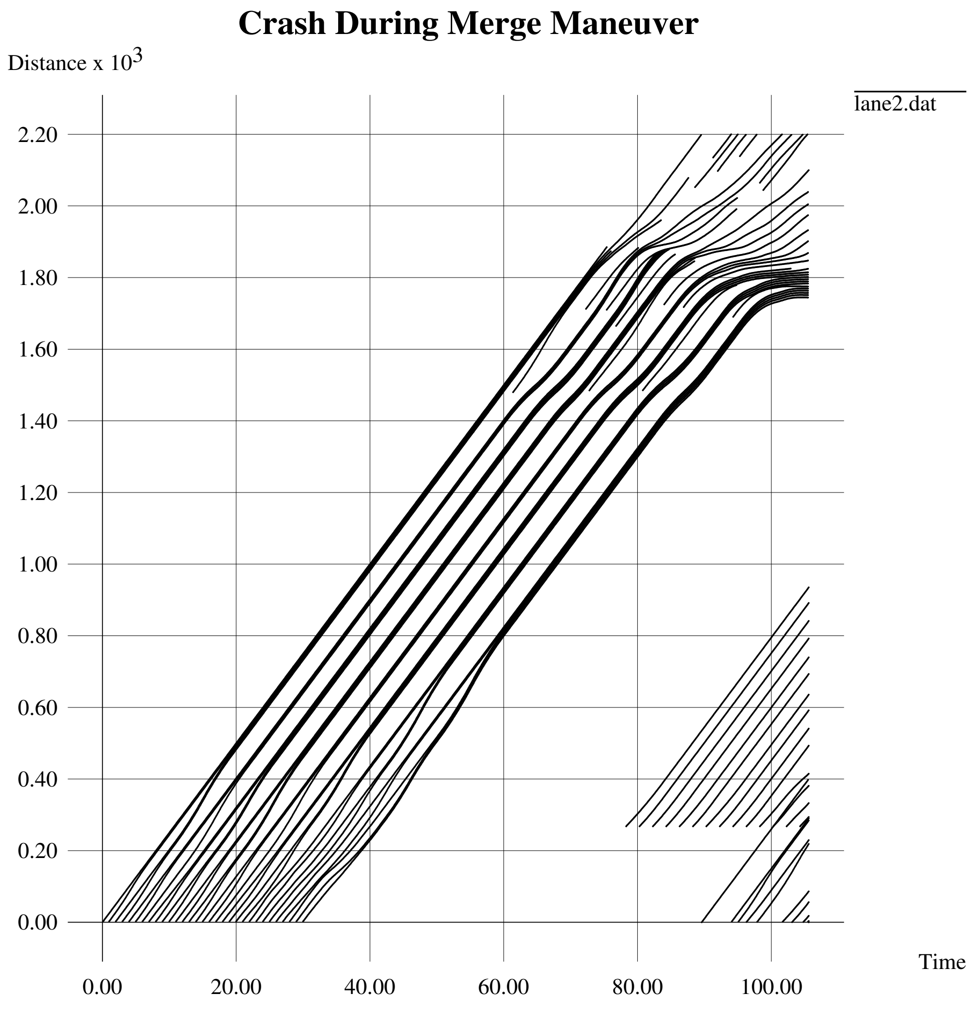

Figure 9: Distance-Time plot of lane 2 showing crashes related to merge maneuvers: there is a four car pile-up at 82 seconds and 1880 meters whose close up is presented in the next plot. There is another crash at 94.8 seconds and 1780 meters

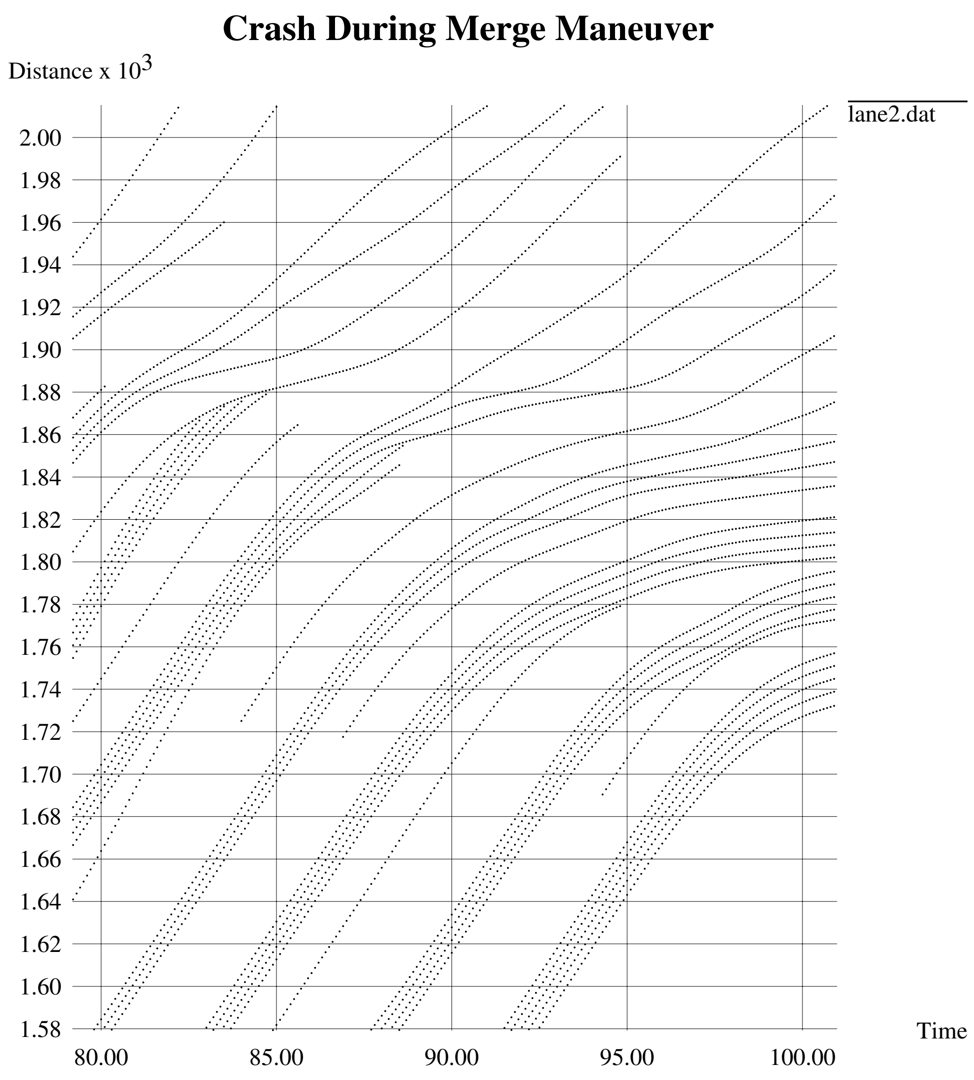

Figure 10: Both the crashes mentioned above can be seen in this plot clearly. The crashes during the merge maneuver occur because of a large deceleration created upstream by the highway junction

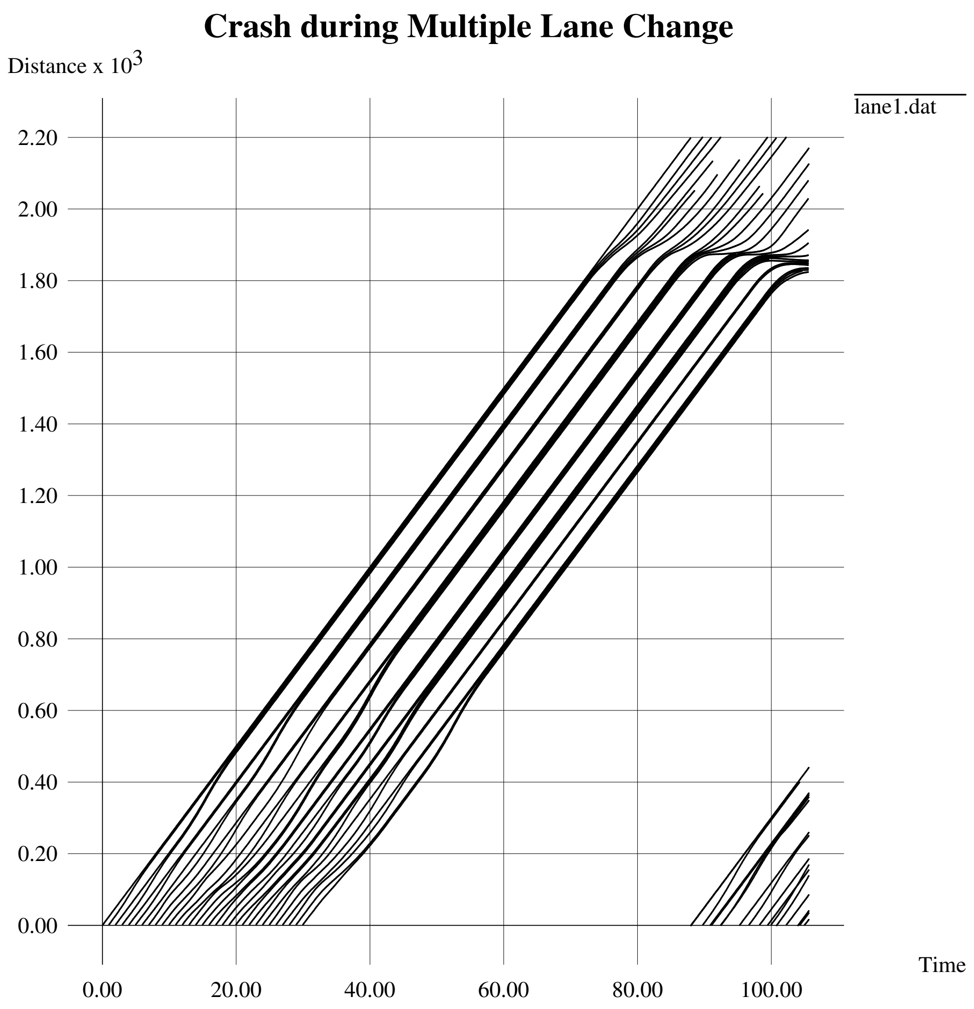

Figure 11: Distance-Time plot of lane 1 showing problems associated with multiple lane change

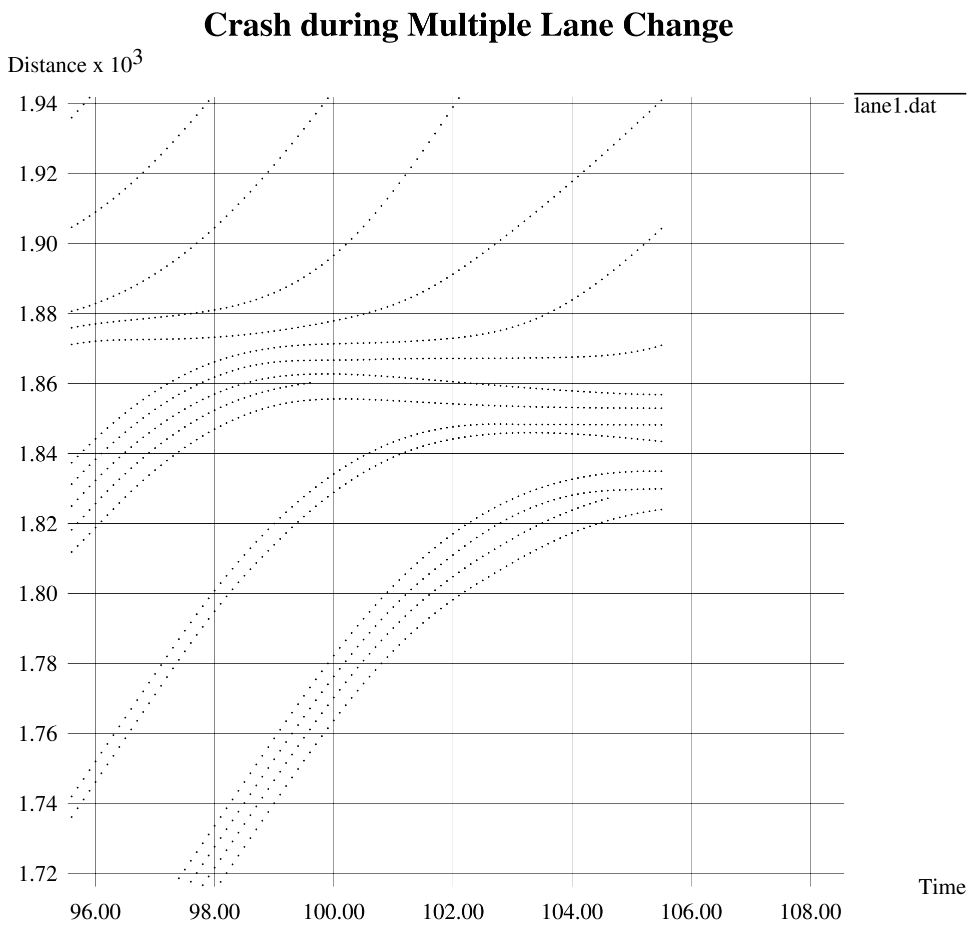

Figure 12: Multiple lane change close up: All three followers of last platoon want to change lane around 101 seconds. All of them split from each other successively and decide to change lane in the same space on adjacent lane. As they are independent platoons, they are not communicating with each other. The third car in this platoon can be seen to crash into the second car at 104.61 seconds and 1825 meters

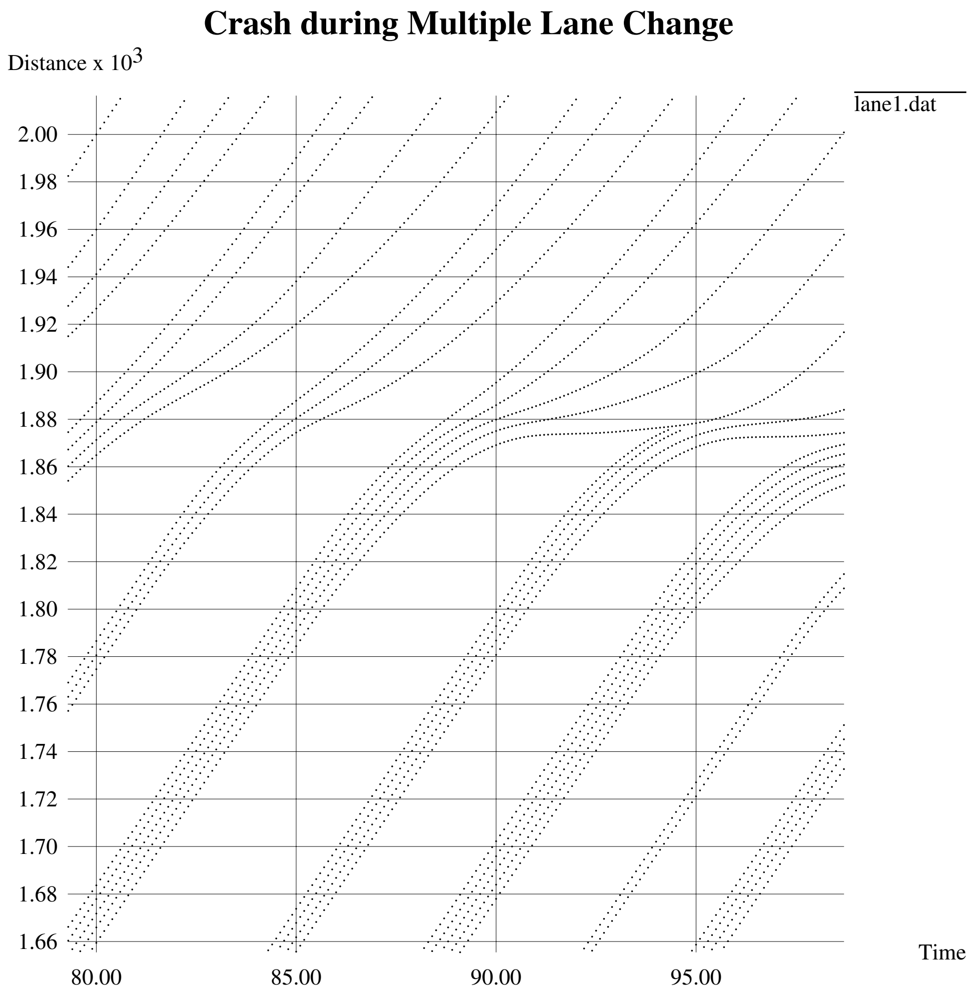

Figure 13: Distance-Time plot of lane 1: crash at 94.31 seconds and 1880 meters is similar in nature to the case of ”changing from a fast to a slow lane”