Towards a Fault Tolerant AHS Design*
Part I: Extended Architecture

John Lygeros, Datta Godbole, Mireille Broucke
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720
lygeros, godbole, mire@robotics.eecs.berkeley.edu

Abstract

We propose a hierarchical control architecture for dealing with faults and adverse environmental conditions on an Automated Highway System (AHS). Our design builds on a previously designed control architecture that works under normal conditions of operation. The faults that are considered in our design are classified according to capabilities remaining on the vehicle or roadside after the fault has occurred. Information about these capabilities is used by supervisors in each of the layers to select appropriate control strategies. We outline the extended control strategies that are needed by these supervisors of each layer of the hierarchy and, in certain cases, give examples of their detailed operation. A companion paper develops and verifies protocols for extended coordination layer strategies and maneuvers.

*Research supported by the PATH program, Institute of Transportation Studies, University of California, Berkeley, under MOU-135

Contents

1 Introduction

Intelligent Vehicle Highway Systems (IVHS) has been an active research area within the California PATH[1] project for the past several years. The objective is to develop an Automated Highway System (AHS) design that will significantly increase safety and highway capacity without building new roads, by adding intelligence to both the vehicle and the roadside. Several approaches to this problem have been proposed within the PATH project, ranging from Autonomous Intelligent Cruise Control (where the driver is in control of vehicle steering) to full automation. An underlying assumption in most of these designs has been that the operation takes place under normal conditions. The definition of "normal" may vary from case to case, but, in general, it means benign environmental conditions and faultless operation of all the hardware, both on the vehicles and on the roadside. Some studies to deal with "abnormal" conditions have been made (for example [1, 2, 3]), but they are mostly concerned with specific faults rather than a general framework. Our goal is to propose an AHS design that will perform well under almost any condition. The only abnormal conditions of operation that we do not consider are faults in the design (e.g., a deadlock in the protocols) and in the implementation of the software, since we assume the design will be verified before being implemented. Even with this restriction it is clear that the task is large. In this report we only give an overview of what is involved and establish a framework for tackling the problem. The framework will partition the task into more manageable parts and formalize the requirements that each of them will need to satisfy.

1.1 Overview of Normal Mode Architecture

Our framework builds on the control architecture proposed in [4, 5] for normal modes of operation. Before presenting the framework we give a brief overview of this design to fix the terminology and notation. The central concept of the architecture is that the AHS control problem is too large to be dealt with by means of a single controller. Therefore a hierarchical control structure is introduced.

The design of the control hierarchy outlined in [4] centers around the notion of "platooning". It is assumed that traffic on the highway is organized in groups of tightly spaced vehicles, called platoons. Intuition suggests that doing this should lead to an increase in the capacity and throughput of the highway; indeed theoretical studies indicate that, if such a scheme is implemented successfully, the resulting highway throughput can be as high as four times the current throughput. Moreover, this will be achieved without a negative impact on passenger safety. By having the vehicles within a platoon follow each other with a small intra-platoon separation (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 to the vehicles and the injuries to the passengers will be minimized. The inter-platoon separation, on the other hand, is large (of the order of 30 meters) so 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 up the freeway.

Clearly implementation of such a scheme will require automatic control of vehicles, as human drivers are not fast and reliable enough to produce the kinds of inputs necessary for forming platoons. In the architecture outlined in [4] the system is organized in five layers (Figure 1). The top layer, called the **network layer**, is responsible for the flow of traffic on the entire highway system[2]. Its task is to prevent congestion and maximize throughput by dynamic routing of traffic.

The second layer, called the **link layer**, coordinates the operation of whole sections (links) of the highway and operates at the roadside. Its primary concern is to maximize throughput while

[1] PATH stands for Partners for Advanced Transit and Highways

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

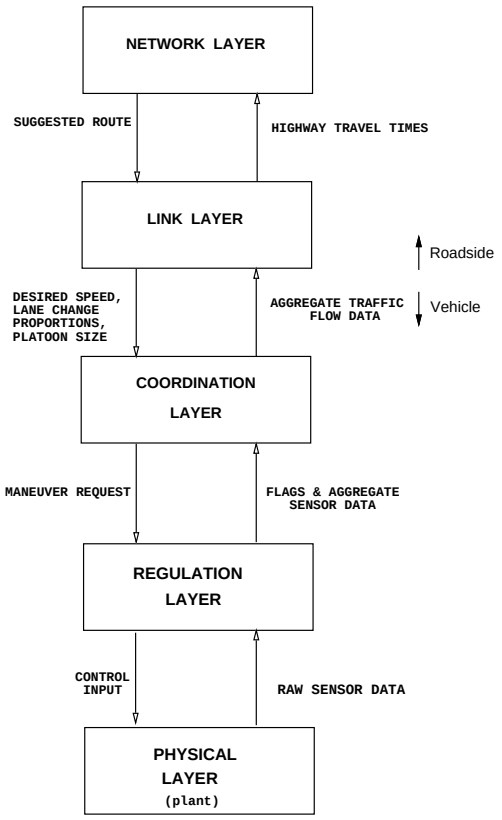

Figure 1: IVHS Architecture

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 treats the vehicles in a section in an aggregate manner rather than considering the state of individual vehicles or platoons. The commands it issues are not addressed to individual vehicles but to all the vehicles in a section; 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". A possible design for the link layer is described in [1]. It is based on a 'traffic "flow" model similar to the ones developed for manual traffic.

The next level of hierarchy below the link layer is the **coordination layer**. It resides in each vehicle and 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 join to form 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 comprise this 30% and split the platoons accordingly in order to let them out. The current design [6] 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 join 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**. Its task is to receive the coordination layer commands and translate them to throttle, steering and braking input for the actuators on the vehicle. For this purpose it utilizes a number of continuous time feedback control laws ([7, 8, 9, 10, 11, 12]) that use the readings provided by the sensors to calculate the actuator inputs required for a particular maneuver. The regulation layer communicates with the coordination layer to inform it of the outcome of the maneuver.

The bottom layer of Figure 1 is not part of the controller. 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).

2 Outline of Proposed Solution

2.1 Design Goals

The references given in the introduction describe models and control strategies that fulfill the requirements set for all the layers of the architecture. All of these laws have been verified theoretically and tested in simulation and, in some cases, in experiments. They have been proven to perform well, mostly under the assumption that the conditions of operation are "normal" (in the sense discussed above). Because the normal mode controllers are typically designed to be robust to external disturbances, the dividing line between "normal" and "degraded" conditions is fuzzy. For autonomous operation we would like the system to be able to deal with a wide range of conditions, significantly wider than the range of robustness of the normal mode controllers. Some laws also exist for operation under severely degraded conditions. For example, [1] contains a link layer control law to divert traffic when a lane is closed (because of an accident for example) and [2] describes a regulation layer control law for steering in case of a tire burst. These degraded mode laws however have been designed to deal with a specific fault and do not provide a general framework for operation under degraded conditions.

Our goal here is to encompass all the laws (for normal and degraded conditions of operation) into a general framework.[3] The result will be a fault tolerant control architecture for the AHS. The new architecture will be a qualitative as well as a quantitative extension of the normal mode architecture. Qualitatively, the degraded mode architecture maintains the hierarchical structure introduced in [4] but it increases the *autonomy* of the system. This is achieved by adding to the design the capability to detect faults, decide on a new control strategy and execute it. Moreover, the extended architecture adds new control laws and maneuvers in each level of the hierarchy, thus quantitatively extending the normal mode architecture.

2.2 General Features

The requirement for increased autonomy and the complexity of the problem imply that the extended design should possess certain features.

2.2.1 Hierarchical Structure

First of all it seems that any fault tolerant AHS design will have to be hierarchical. Even for normal mode operation the complexity of the problem forced the designers to introduce a hierarchical controller. For the degraded mode architecture we have to deal with the additional complications arising from the reduction in the capability of the system, therefore the need for a hierarchy is even more pronounced. The controller presented here will maintain the normal mode hierarchical structure (number of levels, abstraction at each level, etc.).

2.2.2 Information Flow

The normal mode architecture implicitly assumes that the system capability is fixed and known a priori. As a result the only information that the levels of the hierarchy need is sensor data on the current state of the system (including the control messages passed from one level to the other). This assumption is no longer valid, however, under degraded conditions of operation. Therefore the degraded mode controllers need information about the current capability of the system (in addition

[3] To complete the framework it will be necessary to design a number of new control laws to supplement the existing ones.

to the sensor data) to select the best possible action. Moreover this additional information needs to be presented in a form compatible with the abstraction of the controller at each level of the hierarchy.

We propose to extend the information flow by the addition of two more hierarchical structures. The **capability structure** encodes the discrete changes in the system capability due to faults in the vehicle and roadside hardware. The proposed design is in the form of a hierarchy of predicates, i.e., functions that return either "1" (if the system possesses a certain capability) or "0" (if it does not). The **performance structure** encodes the gradual degradation in the system performance due to adverse environmental conditions and gradual wear of the vehicle components. The proposed design consists of a set of maps from the causes of gradual performance degradation to a set of parameters (such as maximum and minimum acceleration, sensor ranges etc.) that reflect the performance of the system. We assume that the sensor structure has already been designed for the normal mode (see for example [13]).

2.2.3 Control Structure

To complete the design we need to specify a controller hierarchy. The closed loop system will then look like Figure 2. The autonomy requirement implies that we need to make an explicit distinction between strategic planning and control execution. Each level of the control hierarchy will be divided into two layers. The top layer is a planner that is responsible for selecting a control strategy which is consistent with the current state of the system. The lower layer is a regulator that is responsible for tracking the strategy commanded by the planner. This distinction was not made explicit in the normal mode of operation, as the control strategy is fixed and the controller is only responsible for its execution (i.e., the planning level is trivial).

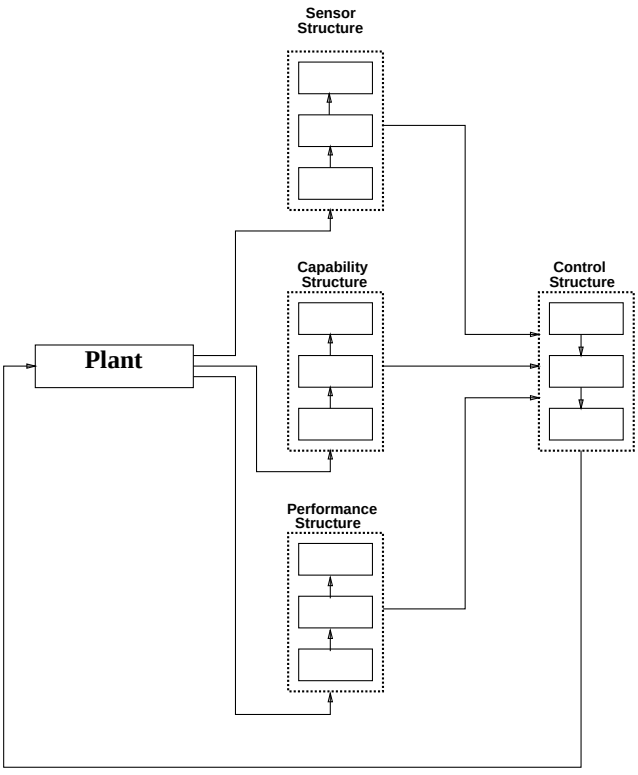

Figure 2: Overview of the Supervision Problem

An issue that needs to be addressed by the controller design is that of optimality. The design of the degraded mode architecture involves a trade off between safety, passenger comfort, performance degradation and design complexity. Ideally we would like to be able to come up with an "optimal" compromise. The formulation of such an optimal control problem is very hard however. Even the fact that we are a-priori restricting ourselves to a hierarchical design implies that the controller will be "suboptimal", as it is unlikely that the optimal solution for any meaningful cost criterion will be hierarchical. In this paper we present a solution to the degraded modes problem, without making any optimality claims. The related questions will be discussed further in Section 6.

2.3 The Design Process

Our approach to the design of a degraded modes architecture involves a number of steps:

1. Identify faults and causes of gradual performance degradation
2. Develop ways of modeling the capability of the system and determine the effect of the factors in 1 on the capability
3. Classify the factors in 1 according to their effect on 2.
4. Extend the control architecture to deal with the classes established in 3
5. Design controllers for the extended architecture
6. Verify (wherever possible) and simulate the extended architecture
7. Identify the shortcomings of the proposed design and return to 4 to fix them

Steps 1, 2 and 3 will be the main topic of this report. For step 1, an exhaustive list of faults and other causes of performance degradation was compiled and is given in the Appendix. For step 2, a framework for modeling the capability of the system in the presence of faults is given in Section 3. The fault classification (step 3) induced by our modeling framework is discussed in Section 4. Based on the work carried out for the first three steps, in Section 5 we discuss the requirements that the extended controllers (step 4) for the link, coordination and regulation layers need to satisfy. We also give a brief description of a possible coordination layer design and state the additional requirements that its implementation imposes on the physical layer. The details of the design, as well as verification results (corresponding to steps 4 and 6) are the topic of a companion report ([14]). It should be noted that many iterations between steps 4 and 6 may be needed before a satisfactory design is obtained.

3 Modeling Capability

3.1 Capability Monitor

The control scheme for normal operating conditions presented in [4] relies on a number of sensors, actuators and communication devices, both on the vehicles and on the roadside. All this additional hardware as well as the standard mechanical parts of the vehicles are prone to failure. Such a failure, in either the vehicle or the infrastructure will directly influence the capabilities of the system as a whole and therefore restrict the controls that the supervisor can implement.

To monitor the capability of the system we propose a design based on a hierarchy of predicates. Each predicate will monitor one functional capability and will return a 1 (True) if the system possesses the capability in question or a 0 (False) otherwise. The predicates will be arranged in a hierarchy similar to that of the normal mode supervisor. The values returned by the higher level predicates will depend on the values of the lower level predicates. This scheme can be used to systematically go through combinations of faults and design specialized control laws that utilize the remaining capabilities so that the impact of the faults on the system is minimized in each case. We will start describing this hierarchy at the bottom and work our way up.

3.1.1 Physical Layer Predicates

The supervisor structure assumes that the vehicles and the roadside have access to certain resources, namely, sensors, actuators and communication devices. We model each one of these resources as a predicate, that returns 1 if the resource is available and functioning and 0 otherwise. Assuming that the supervisor requires n_a actuators, n_s sensors and n_c communication devices, the capability of the physical layer can be expressed as a vector of zeros and ones of dimension $n_s + n_a + n_c$:

$$\{0,1\}^{n_s+n_a+n_c}$$

This vector reflects which resources are functioning and which are not. It should be noted that for simplicity the actuator predicates are interpreted as reflecting the capability of the vehicle to accelerate, decelerate and turn. Therefore they incorporate information about basic vehicle functionality, like engine and tires being in proper working order, enough fuel, etc. Predicates for these basic functionalities can explicitly be added at the cost of a small increase in the complexity of the monitor.

3.1.2 Regulation Layer Predicates

The regulation layer contains a number of control laws, both longitudinal and lateral. Each one of these laws makes use of a number of physical layer resources, primarily sensors and actuators. For a regulation layer controller to be functional, all of these resources need to be available. Therefore, the applicability of a regulation layer controller can be modeled by a predicate whose value depends on the values of the predicates for the physical layer.

Consider, for example, the Autonomous Intelligent Cruise Controller proposed in [10] as the default longitudinal law for the leader of a platoon. This longitudinal control law uses sensor readings of velocity and acceleration of the vehicle, and of the spacing and relative velocity with respect to the preceding vehicle to calculate inputs for the throttle and brake actuators. Without getting into the details of the control law, we can see that the lead controller predicate can be viewed as an *AND* predicate on the values returned by the predicates for the velocity, acceleration, spacing and relative velocity sensors and the brake and throttle actuators. Likewise, the law proposed in [7] for the followers in a platoon makes use of additional information about the state of the leader of the

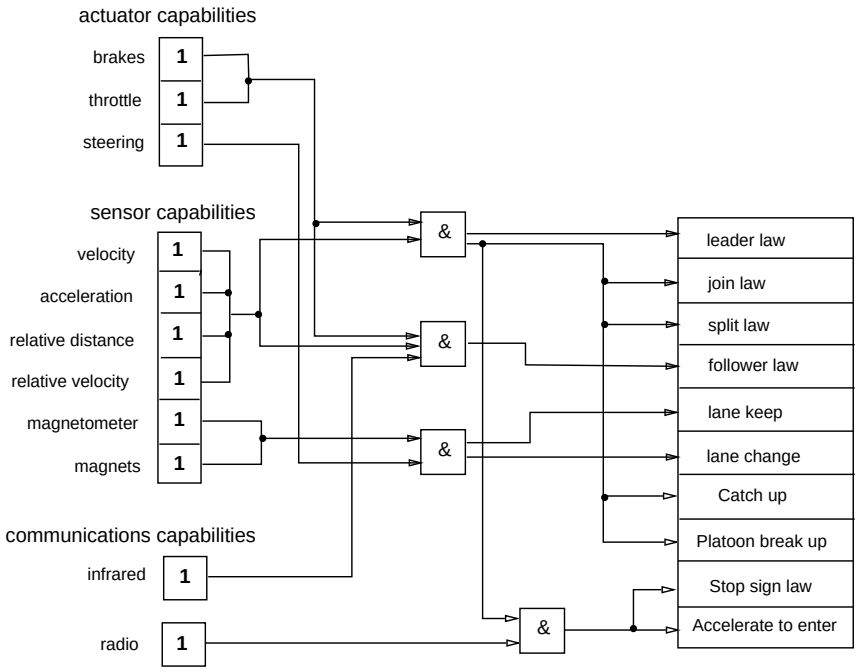

Figure 3: Connection between Physical and Regulation layer capabilities

platoon. It is assumed that this information will be transmitted to all the followers using an infrared communication link. Therefore, the predicate for the longitudinal follower law should depend on the predicate for the infrared communication link (as well as the predicates for the sensors and actuators listed above).

In this formalism the capability of the regulation layer can be encoded by a vector of zeros and ones, of dimension equal to the number of control laws available to the layer. If there are n_{long} longitudinal laws and n_{lat} lateral laws this vector will have the form:

$$\{0,1\}^{n_{long}+n_{lat}}$$

As shown in the examples, the design of the control laws implies a mapping from the vector coding the capabilities of the physical layer to the vector coding the capabilities of the regulation layer:

$$F_R : \{0,1\}^{n_s+n_a+n_c} \longrightarrow \{0,1\}^{n_{long}+n_{lat}}$$

This mapping is easy to convey by means of a figure. Figure 3 shows the mapping for the control designs in [8, 9, 10, 11, 12].

3.1.3 Regulation Layer Supervisor Capability Predicates

From the point of view of the coordination layer, the regulation layer control laws represent resources that can be used to carry out maneuvers. Typically, each maneuver will need to make use of two control laws, one longitudinal and one lateral. Therefore, in order for the coordination layer to be able to invoke certain maneuvers, the relevant control laws should be operational. For example, for the coordination layer to command a platoon leader to join, at least one (of possibly many) longitudinal join law and a lateral lane keeping law should be operational.

The capabilities of the regulation layer, when seen from the point of view of the coordination layer, can be modeled by predicates that depend on the regulation layer capability vector.

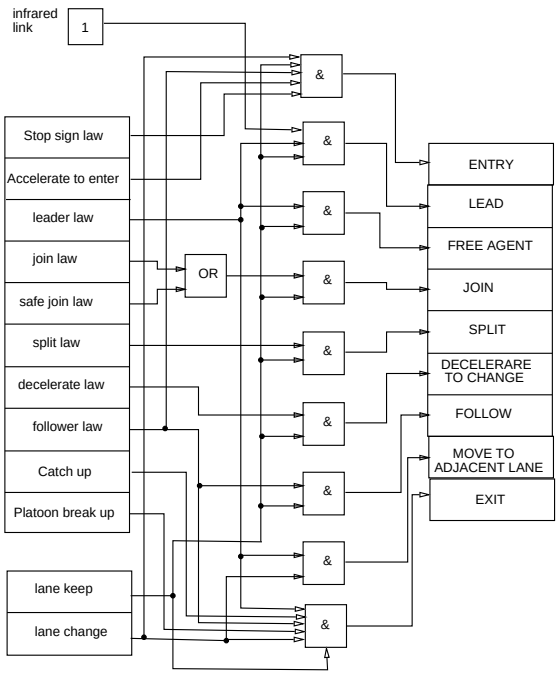

Figure 4: Regulation layer supervisor capabilities

These predicates form a regulation layer supervisor capability vector. Let n_{man} denote the number of maneuvers that may be requested by the coordination layer. Then the regulation layer supervisor capability vector will be a vector of zeros and ones of dimension n_{man}. The design of the supervisor induces a mapping between the capability vectors of the regulation layer and its supervisor.

$$F_I : \{0,1\}^{n_{long}+n_{lat}} \longrightarrow \{0,1\}^{n_{man}}$$

For the normal maneuvers presented in [6, 11] and the control laws of [8, 9, 10, 12, 15], the map F_I can be seen in Figure 4.

3.1.4 Coordination Layer Supervisor Predicates

In order to operate normally, the coordination layer of [6] requires the vehicle to be able to perform certain maneuvers. More specifically a normal vehicle should be able to enter the AHS, lead a platoon, be a follower, join to a platoon (provided it is a platoon leader), split from a platoon (provided it is a follower), decelerate to facilitate a lane change, move from one lane to another and exit from the AHS. As discussed in the previous section, the capability to carry out these maneuvers will be coded by the regulation layer supervisor capability vector. In addition, to execute the protocols that organize the maneuvers, the coordination layer needs access to certain communication capabilities. Therefore, whether the coordination layer can operate in its normal mode or not can be expressed as a predicate on the values of the capability vectors for the regulation layer supervisor and the communications. A fault in the physical layer that will damage any one of these capabilities will render the normal mode coordination layer inoperable. For this reason alternative coordination layer protocols that are still operational under reduced capabilities will have to be designed.

An example of such a set of protocols is the *Take Immediate Exit* strategy described in Section 5.2. The objective of this strategy is to take a vehicle that has developed a fault (and therefore can not function in the normal mode) out of the highway as soon as possible. The design makes

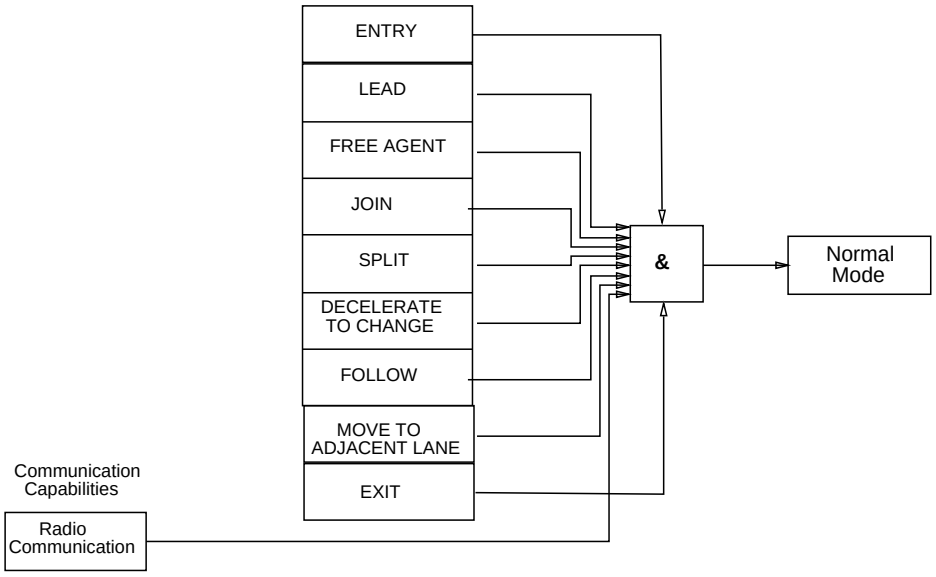

Figure 5: Coordination layer supervisor capabilities

use of additional maneuvers, which can easily be added to the predicate structure for the regulation layer and its supervisor. As we shall see in Section 5.2, some of these maneuvers will require close cooperation with the neighboring vehicles. Therefore, the applicability of the *Take Immediate Exit* strategy for the coordination layer can be expressed as a predicate on the values of the regulation layer supervisor capability vector, the communication capability vector of the faulty vehicle and the regulation layer supervisor capability vectors of the neighboring vehicles. It is assumed that knowledge about the capabilities of the neighbor will be obtained once communication has been established.

Once many such strategies have been designed (see Section 5.2) the coordination layer capability can be expressed as a vector of zeros and ones. The dimension of this vector will be equal to the number of these strategies, which will be denoted here by n_{coord}. The design of the strategies induces a mapping:

$$F_C : \{0,1\}^{n_{man}} \times \{0,1\}^{n_c} \times \{0,1\}^{N n_{man}} \longrightarrow \{0,1\}^{n_{coord}}$$

Here N stands for the maximum number of neighboring platoons that may need to cooperate in an emergency maneuver. The part of F_C dealing with normal operation is shown in Figure 5. Similar maps are contained in Appendix C for the strategies introduced to deal with degraded conditions of operation.

3.1.5 Link Layer Supervisor Predicates

As discussed in the introduction, the link layer design of [1] makes use of information about the density and average velocity of traffic in a link to come up with control inputs that will maximize the throughput of the highway. In order to improve the resolution of the information and the commands, the link is partitioned into smaller sections. Each section consists of a single lane and its length is (typically) smaller than that of the link. Even though the controllers we propose for the link layer operation under faults are quite a bit different (see Section 5.1) we still maintain this partitioning of a link into sections in our capability structure.

In addition to density and average velocity information (which will be provided by the sensor hierarchy), a link layer design for degraded conditions of operation needs information about discrete events that limit the capabilities of its sections. The example presented in Section 5.1 indicates four such events: section is blocked, section contains no vehicles, section contains vehicles queued behind an accident and section contains emergency vehicles. Similar events are also relevant for any entrances and exits that may be contained in the link.

These properties can be modeled as a set of predicates for each section, entrance or exit that return one if the link possesses the property (e.g., is blocked) and zero otherwise. The value returned by these predicates should depend on the capability vectors of all vehicles in the section. For example, if a section contains a broken down vehicle then the predicate for "section is blocked" should return one. In addition the values of the predicates should also reflect certain infrastructure faults. For example the fault "uncontrolled object in the lane" (which may refer to debris from an accident in one lane spilling over to an adjacent lane) should also cause the predicate "section is blocked" to return one. Let n_I denote the number of the relevant infrastructure faults, N_i the number of platoons in section i and n_{sec} the number of predicates for each section ($n_{sec} = 4$) then for all sections, entrances and exits contained in the link we can define maps:

$$\begin{aligned} F_{s_i} &: \{0,1\}^{N_i n_{coord}} \times \{0,1\}^{n_I} \longrightarrow \{0,1\}^{n_{sec}} \\ F_{en_j} &: \{0,1\}^{N_j n_{coord}} \times \{0,1\}^{n_I} \longrightarrow \{0,1\}^{n_{sec}} \\ F_{ex_k} &: \{0,1\}^{N_k n_{coord}} \times \{0,1\}^{n_I} \longrightarrow \{0,1\}^{n_{sec}} \end{aligned}$$

where i, j, k range over the number of sections, entrances and exits contained in the given link. As will be seen in Section 5.1, the values of the output predicates for each section will be used by the link layer controllers as events that trigger transitions from one desired velocity and density profile to another.

3.2 Performance Monitor

The performance monitor is designed to continuously determine the effect of disturbances on the system and draw the line between acceptable and unacceptable degradation of performance. In case of acceptable disturbances, the controller parameters can be tuned (on-line) to improve system performance. The performance monitor invokes a degraded mode controller at the occurrence of an unacceptable disturbance.

3.2.1 Robustness Analysis Framework

There are three elements involved in this process. The first is the causes of gradual performance degradation, which the supervisor will have to guard against. They include adverse weather conditions (such as rain, fog or snow) and gradual hardware degradation (such as brake ware). An extensive list, compiled by consulting with numerous PATH researchers is given in Appendix B. We will use $\mathcal{C}$ to denote the set of performance degradation causes. Assuming there are c such causes, $\mathcal{C}$ has the form:

$$\mathcal{C} = \{C_i / i = 1, \ldots, c\}$$

Each C_i is a real number whose magnitude signifies the severity of the cause (e.g. the longitudinal wind measured in meters per second). [4]

[4] In its simplest form C_i can be thought of as a predicate that returns 1 if cause i is present and 0 if it is not. "Soft" approaches, such as fuzzy logic may be used to quantify more elusive causes, such as snow or fog.

The second factor is the performance parameters that can be used to monitor the capability of the system. These performance parameters depend on the layer of the architecture and include, for example, the maximum and minimum deceleration available to the vehicle (for the physical layer), and the maximum tracking error of the various continuous time controllers (for the regulation layer). We use $\mathcal{P}$ to denote the set of performance parameters. The set $\mathcal{P}$ can be divided according to the level of the hierarchy associated with each parameter.

$$\mathcal{P} = P_P \cup P_S \cup P_R \cup P_C \cup P_L \cup P_N$$

where $P_P = \{P_P^i, \quad i = 1, \ldots, n_p\}$ are the parameters associated with the physical layer, $P_S = \{P_S^i, \quad i = 1, \ldots, n_s\}$ are the ones associated with the sensors and communication devices, $P_R = \{P_R^i, \quad i = 1, \ldots, n_r\}$ the ones associated with the regulation layer, etc. A list of the associated performance parameters considered for each level is given in the appendix.

The final factor is the performance requirements. They can be thought of as bounds on the performance parameters. More formally performance requirements can be thought of as predicates on the space of performance parameters:

$$R_i : \mathcal{P} \longrightarrow \{\text{True}, \text{False}\} \qquad i = 1, \ldots, r$$

Robustness analysis involves finding functional relationships between causes of gradual performance degradation and the performance parameters. In other words we would like to establish a map:

$$f : \mathcal{C} \longrightarrow \mathcal{P}$$

that determines how the causes of performance degradation affect the performance parameters. This map will depend on the details of the control laws. It can be quantitatively altered by changes in the controller parameters. The qualitative dependencies will be fixed, unless major changes are made in the design. The qualitative dependencies for the set of performance parameters discussed above are outlined in the appendix. The range of conditions $\hat{\mathcal{C}}$ under which the performance of the system is acceptable is now given by:

$$\hat{\mathcal{C}} = \bigcap_{i=1}^{r} f^{-1}(R_i^{-1}(\text{True})) \subset \mathcal{C}$$

Many iterations (off-line) may be needed in order to properly capture the system requirements in terms of the above equation for $\hat{\mathcal{C}}$.

3.2.2 Robustness Enhancement

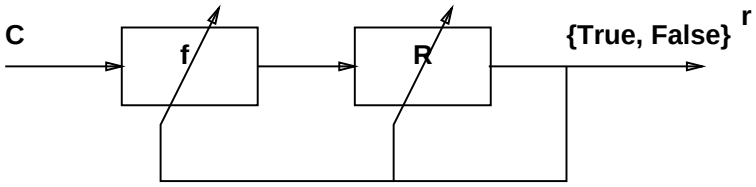

Figure 6: On-line controller tuning

Enhancing the robustness of the system involves enlarging $\hat{\mathcal{C}}$. This framework can be used for off-line robustness enhancement of the design, where the controllers are tuned (off-line) to accommodate the largest set of conditions $\hat{\mathcal{C}}$. The framework can also be used to increase the system autonomy by

on-line tuning of the controllers. If the requirements of the control laws are not met by the capability parameters at any time (e.g., a join controller may need the vehicle to decelerate faster than the capability of the vehicle at the current time), the control laws are tuned until the new requirements are met by the parameters. This process is represented by Figure 6.

3.2.3 Degraded Mode Initiation

Even after the domain $\hat{\mathcal{C}}$ has been maximized, there will probably still be some conditions in $\mathcal{C}$ which are not covered. These conditions for which performance is unacceptably degraded will be treated by the supervisor in a way similar to the treatment of loss of capability due to faults. In this sense, the effect of gradual degradation and limits of robustness can be modeled as an extra term on the predicates (Figure 7). Overall the degraded mode controllers will have to be designed for the causes

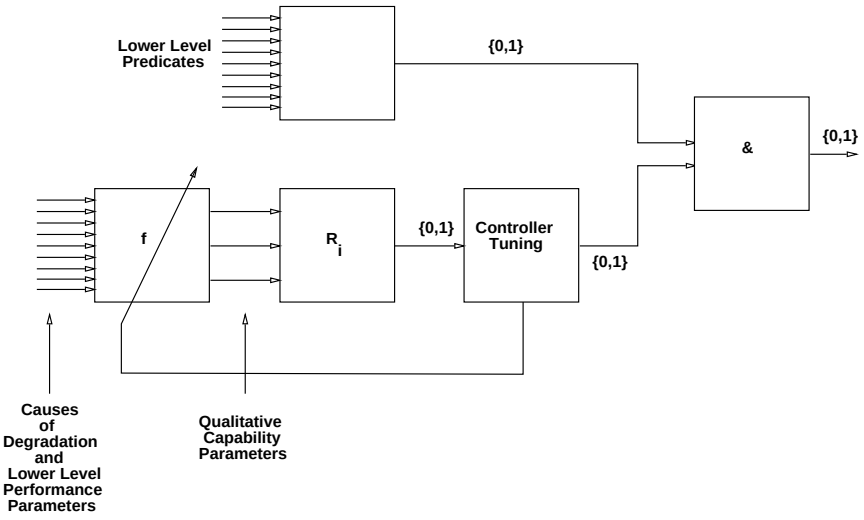

Figure 7: Introduction of robustness predicates

in $C - \hat{\mathcal{C}}$.

3.2.4 Examples

Based on the normal mode AHS design, we have identified the causes of performance degradation and the performance parameters. Appendix B contains the list of causes of gradual performance degradation. The list is further classified into environmental causes and vehicle causes. Note that this list also depends on the the particular design of the AHS, which in this case is the platooning architecture of [4]. In Appendix B, we provide a list of performance parameters classified according to the hierarchical structure of the controller. We have also identified the qualitative composition of the maps f that capture the effect of $\mathcal{C}$ onto $\mathcal{P}$. The complete quantitative specification of the maps will be possible after testing the system in the aforementioned environment. Once these maps are determined, the performance monitor along with the controller tuning module can be completely designed. We outline the possible design of the robustness enhancement module by examples:

Leader Control

We suggest some simple changes that would increase the autonomy of the current leader control laws.

1. If the limits on P_P^1 are violated, the join/split/change lane trajectories used in the regulation layer feedback control design [10] should be recalculated to accommodate for the reduced acceleration capability. Different limits for aborting the maneuvers should be set in the interface of [16].

2. If the limits on P_P^2 are violated similar measures should be taken. In addition the headway (inter-platoon distance) should be increased and/or the desired speed should be reduced. For example, suppose

$$\begin{aligned} R_1 &= \{a_{min} < -3\} \\ R_1' &= \{a_{min} \in [-3, -1)\} \\ R_1'' &= \{a_{min} \geq -1\} \quad = \text{Boundary of unacceptable } a_{min} \end{aligned}$$

   the normal mode requires the deceleration to be at least $-3m/s^2$. If some C_i (say rain) causes a_{min} to become greater than -3, but less than -1, the robustness module augments the control laws (say changes the join trajectory), inducing a change in f. The corresponding requirement (R_1') is also modified to reflect the change. If some other C_i (say leak in brake fluid) causes a_{min} to become greater than -1, the robustness module calls for a degraded mode.

3. The join control law has to be robust to handle the effect of C_1, C_{14}, C_{17}.

4. If C_{24} affects the sensor range, then the desired speed should be reduced and/or the headway should be increased.

Similar measures can be taken for the remaining control laws. For example for a lateral controller, if the limits on P_R^2 are violated the desired speed should be reduced, particularly on a curved road. It should be noted that the three step procedure suggested here (causes of degradation, performance parameters, performance requirements) allows us to simplify the robustness analysis task. In the above discussion only the performance parameters were needed to tune the controllers. Effectively we were to group large numbers of performance degradation causes into classes, based on their effect on the performance parameters. Apart from the obvious simplification this also makes the design a lot more flexible, as it allows us to add the effect of more causes of degradation with minor changes, by linking them to the performance parameters.

4 Fault Classification

Appendix A contains a comprehensive list of faults, for both the vehicle and the infrastructure. Because the list is so large, we would like to be able to design controllers that deal with whole classes of faults or combinations of faults. In this section we show how the capability and performance structures can be used to induce such a classification. The classes reflect the potential available in the system in the presence of a combination of faults and adverse conditions. Faults in the same class will lead to the same predicates in the capability structure returning zeros. Using this principle as a guide we are able to distinguish the following classes:

Vehicle stopped, must stop: This class contains the most serious vehicle faults. The vehicle can not continue moving on the AHS safely and has either already come to a stop or it should be commanded to do so and wait to be towed away. Because of the severity of the situation, all the layers of the control architecture will undergo some degradation in performance and assist in resolving the fault condition.

Faults in this class will typically lead to a false "Capable of being a free agent" predicate in the regulation layer supervisor. This will in turn lead to predicates returning zeros all the way up to the link layer. Depending on the type of fault we identify three subcategories which are differentiated by the technique that is used to stop the faulty vehicle. The subclasses and the faults contained in each one of them are listed in the appendix.

Vehicle needs assistance to get out: The faults in this class are slightly less serious. The vehicle may continue moving but has lost some essential capability and it must therefore exit the AHS as soon as possible. Moreover, it needs the assistance of its neighbors to do so. Typically, faults in this class will result in the normal mode coordination layer predicate returning a zero without any of the link layer predicates being affected. Therefore, these faults can be handled locally and need not involve the higher levels of the architecture (link and network). As before, the faults are divided into subclasses according to the affected capability.

Vehicle needs no assistance to get out: The faults in this class are even less serious. Typically the vehicle is fully functional but should leave the system soon to avoid further problems and hazards (in case a second fault occurs for example). Typically faults in this class result in regulation layer predicates returning zeros, without any coordination layer predicates being affected. They are handled by special controllers in the regulation layer and neither the neighboring vehicles nor the roadside need to be alerted.

Vehicle does not need to get out: This class contains minor faults that require no special action but should nonetheless be recorded and the driver should be notified in case he needs to alter the travel plan. They result in only physical layer predicates returning zeros.

Infrastructure failures: This class includes all faults that induce a reduction in the capability of the infrastructure. They usually lead to severe degradation in performance. Some of them can be handled by the normal mode controllers of the link and network layers, but some may need drastic changes in the operation of the system. The faults reflected in the infrastructure predicates discussed in Section 3.1.5 are contained in this class. They result in link layer predicates returning zeros, without any changes in the coordination layer predicates.

Driver–Computer interaction down: Problems in this class mainly occur during the entry and

exit to the system. We assume that once on the freeway, the driver may not interfere with the system operation and therefore can not induce any special faults. These faults are resolved by simple additional strategies that do not interfere with the rest of the design (see [11] for details).

Now each of the faults can be assigned to a unique class (the classification was already carried out in Appendix A). It should be noted that, even if the list in the appendix is not exhaustive, any additional faults we come up with can be uniquely classified using this scheme. Moreover combinations of faults can also be classified similarly.

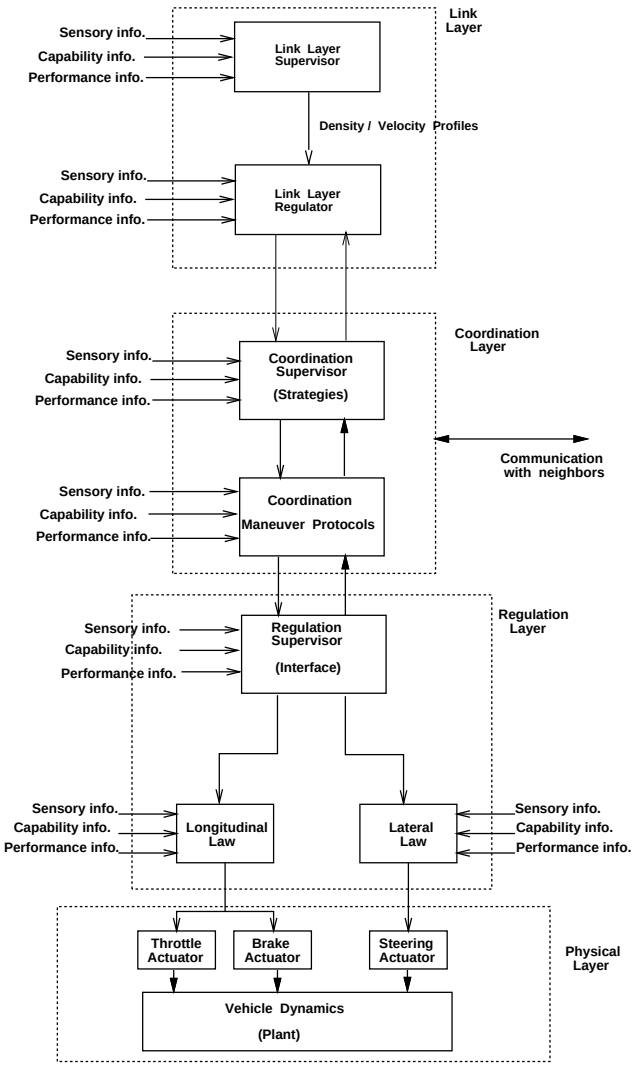

Figure 8: Extended Architecture for Degraded Modes of Operation of AHS

5 Control Design

The extensions discussed so far (i.e., the capability and performance monitors) have been quite general and will be needed in any architecture that is capable of dealing with degraded conditions of operation. In this section we present specific suggestions for controller designs that deal with each one of these faulty conditions. As already discussed in Section 2, controllers at each layer of the control hierarchy consist of two levels, a supervisory level that selects an appropriate strategy and a regulator level which executes individual maneuvers to track this strategy. This distinction is implicitly present in the normal mode architecture. Because there is only one fixed strategy for the normal mode, the division is not explicitly stated.

As part of the extended architecture, we design new strategies at the supervisory level and new controllers for the regulator level for each layer of the control hierarchy. Based on the available capabilities of the roadside and the vehicle, the supervisor selects control strategies in order to respond to the fault. The supervisor uses one of two schemes to select the control strategy. If the fault is vehicle borne, the capabilities maps directly to a coordination layer control strategy, using the

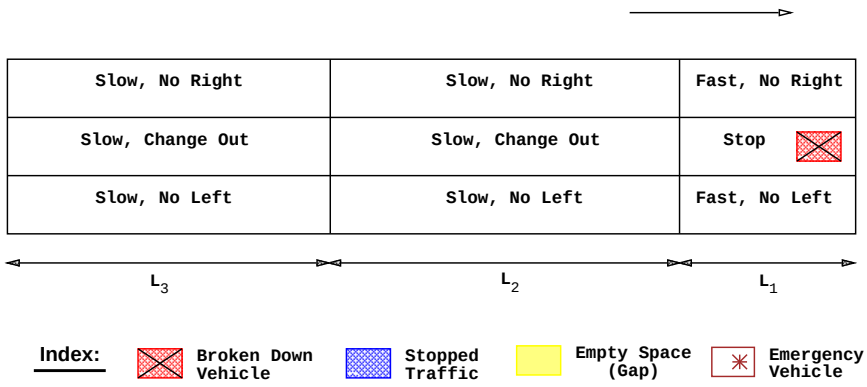

Figure 9: Stage 1: Vehicle stopped on highway

predicate hierarchy of Section 3.1. If the fault is in the infrastructure, or the fault requires assistance from the link layer, the capabilities of the link layer are also affected. In this case, however, a map from infrastructure capabilities to control strategies is harder to obtain, because the model for the link layer is not a discrete event system. As the current design of the link layer uses a flow model, a translation is made from capabilities to control strategies for the link layer using a density/velocity profile generator, which is described in Section 5.1.

What is presented in this section is an outline of control strategies that are proposed for dealing with the fault classes presented in the previous section. These control strategies have not been *explicitly* optimized for capacity and safety. Some discussion of the optimality of the design is given in Section 6.

5.1 Link Layer

5.1.1 Overview

For normal operation, the primary consideration of the link layer is to maintain a smooth flow of traffic and ensure that all vehicles make their exits. Under degraded conditions, however, other considerations such avoiding an incident, facilitating emergency vehicle access, etc. become prominent. To highlight this point consider the following example:

Suppose a faulty vehicle has stopped in the middle lane of a three lane highway. The vehicles immediately behind the faulty vehicle will stop and form a queue. The safety critical task of stopping vehicles upstream before they hit the stopped vehicle is carried out by the regulation layer control laws. The link layer controller will invoke an incident avoidance control strategy. The following figures present snapshots of desired traffic patterns and link layer commands over different time intervals. They indicate both the temporal and the spatial extent of the performance degradation. In Stage 1 (Figure 9), the section labeled stop has a change of capability of the form "section is blocked". Adjacent lanes are slowed down to facilitate the vehicles from the stopped lane to change out. Some vehicles will queue behind the incident.

In Stage 2 (Figure 10), there are no vehicles in the stopped lane in section L_2. There is a gap created in an adjacent lane which travels towards the stopped vehicles at the speed of that lane. This strategy has been triggered by the predicate "section contains queued vehicles" becoming true for section L_1, as well as the predicate "section contains no vehicles" for section L_2 becoming true.

As the gap approaches (Figure 11), the queued up vehicles *Back Up* in the empty space in L_2, speed up to adjacent lane speed and change lane into the gap. The gap creation and vehicle

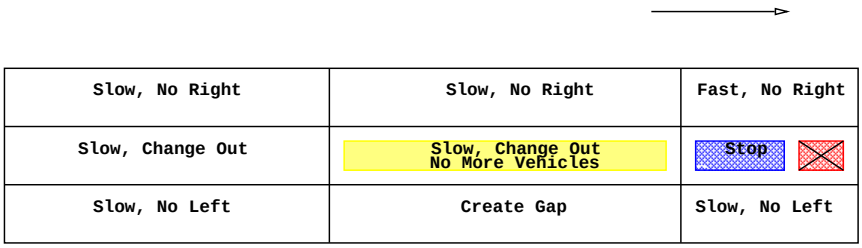

Figure 10: Stage 2: Vehicle stopped on highway

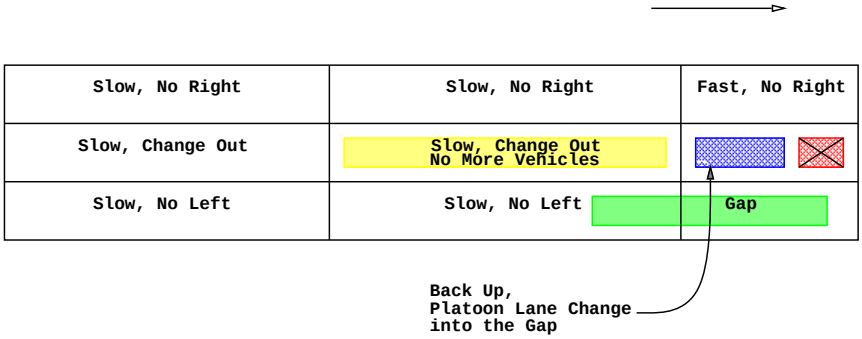

Figure 11: Stage 3: Vehicle stopped on highway

removal will go on until all the vehicles which are queued up behind the faulty vehicle are cleared. This strategy ends when the predicate 'section contains queued vehicles" for section L_1 returns false.

In the meantime (Figure 12), the emergency vehicles move towards the incident using the blocked lane. As the lane is empty from L_2 onwards and the vehicles in this lane in L_3 are moving out, the emergency vehicle will probably be moving faster than the vehicles in adjacent lanes. Alternatively, if the emergency vehicle shows up earlier, then we can stop lane changes from blocked lane to one of the adjacent lanes and let the adjacent lanes carry the emergency vehicle faster. This control strategy is triggered by a change of the capability predicate "section contains emergency vehicles".

In Stage 5 (Figure 13), the emergency vehicle has reached the stopped vehicle and is moving ahead of it (and any remaining queued up vehicles) using the algorithm of Stages 2 and 3 (gap creation in adjacent lane).

Finally, in the recovery mode (Figure 14) some restrictions on speed and lane changing activity are imposed to avoid further crashes due to large velocity differentials across lanes. At this stage all the link layer predicates have returned to their normal mode values.

It should be noted that the proposed strategy is such that the link layer is not a safety critical subsystem. The automated vehicles possess sufficient on-board intelligence to avoid being involved in a catastrophic collision. The role of the link layer is to simply ease the congestion caused by the presence of the fault. Therefore, no further accidents will be caused if the link layer fails to perform its task; the worse that can happen is an increase in travel times.

5.1.2 Link Layer Supervisor

The above example indicates that the control strategy employed by the link layer to clear an incident involves a sequence of discrete steps. Transition from one step to the next is triggered by a change in the link layer capability predicates. The easiest way to visualize this strategy is through a sequence of

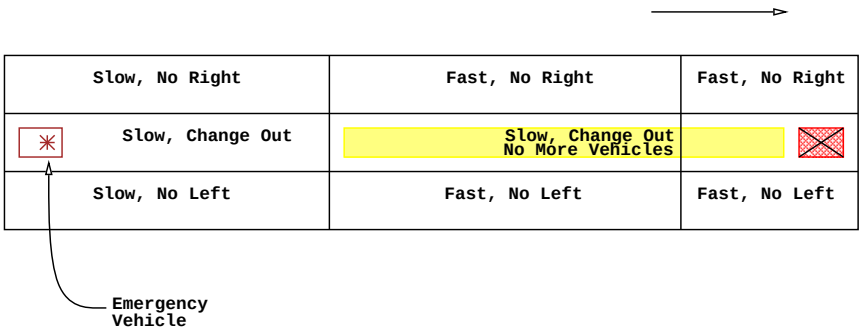

Figure 12: Stage 4: Vehicle stopped on highway

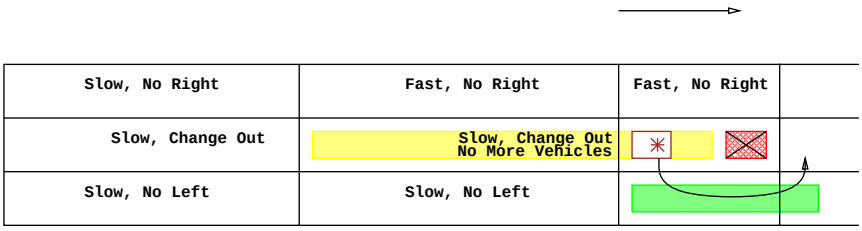

Figure 13: Stage 5: Vehicle stopped on highway

desired density and velocity profiles of the vehicles in the given link. For example Stage 2 corresponds to a velocity profile with zero velocities in lane 2, sections L_1 and L_2, small velocities in lanes 1 and 3, sections L_2 and L_3 and lane 2, section L_3 and large velocities in lanes 1 and 3, section L_1. The corresponding density profile has high densities everywhere, except lane 3, section L_2 and lane 2, section L_3, where the density is low and lane 2, section L_2 where the density is zero (Figure 15).

The link layer strategy can be implemented as a sequence of such profiles. The task of calculating the strategy is carried out by the link layer supervisor, which uses a traffic flow model to produce the sequence of desired traffic patterns. It should be noted here that the description of the strategy need not be in terms of density and velocity profiles. Such profiles are better suited if flow traffic models, such as the ones in [1, **?**], are used. For activity and work based models (like the ones in [**?**, 17]) description in terms of a desired distribution of activities may be more suitable. It should be possible to move from one description of traffic patterns to the other, provided the formalisms are of comparable power.

Implementation of degraded mode strategies, like the one described in the example is

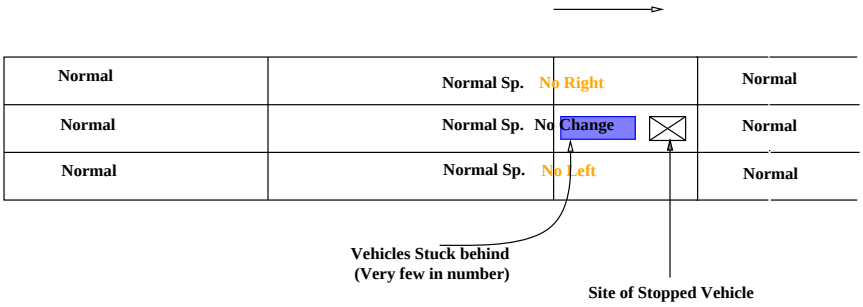

Figure 14: Recovery after the stopped vehicle is towed away

currently underway for the activity model of [17].

5.1.3 Link Layer Regulator

The task of tracking the strategy determined by the supervisor is carried out by the link layer regulator. The objective at this level is to translate the desired traffic patterns to commands for the vehicles in each section. The commands will in general take the form of velocity and lane change suggestions, and join and split suggestions. The nature of the commands will depend on the modeling formalism used by the regulator. Such differences will be taken care of by an appropriate design of a link-coordination layer interface. The regulator should monitor the traffic in the link and use feedback to guarantee that the desired strategy is tracked by the closed loop system.

A possible regulator design is presented in [18]. The design accepts as inputs velocity and density profiles and issues velocity commands to the vehicles (both longitudinal and lateral). Asymptotic tracking of the desired profiles was proved based on a flow model for the traffic and using Lyapunov stability analysis. The performance of the regulator is currently being investigated on the SmartPath simulation platform.

5.2 Coordination Layer

5.2.1 Overview

Similar to the link layer, the coordination layer consists of a two level control structure (Figure 8). The upper level, called the coordination supervisor, is responsible for strategic planning. It determines the sequence of maneuvers that a vehicle should carry out. The lower level contains protocols for coordination of individual maneuvers with the neighbors. We call this level the coordination layer maneuver level. The normal mode coordination layer is structured in a similar way. New strategies are added both to the coordination supervisor and to the coordination maneuver level in order to extend the design to faulted conditions.

5.2.2 Strategies

The coordination supervisor contains a set of *strategies* corresponding to the fault classes of Section 4. A strategy consists of a predefined sequence of atomic maneuvers. For faults in the class "vehicle stopped/must stop" a two step strategy is employed. In the first step, a strategy for stopping the vehicle is chosen while, the second step determines what needs to be done once the vehicle is stopped. If the vehicle is stopped before the fault is detected only the second step is relevant. The strategy employed for the first step depends on which subclass the fault belongs to. If the faulty vehicle has lost its braking capability, then it uses *Aided Stop* strategy in which the vehicle in front of the faulty car applies gentle braking to bring both the vehicles to stop. If the faulty vehicle is a leader, then it executes a *Front Dock* maneuver to become a follower. For other subclasses, the faulty vehicle employs either a *Gentle Stop*, or a *Crash Stop* strategy. The names reflect the severity of braking employed to bring the vehicle to a stop. Once the vehicle comes to rest, the link layer employs strategies to ease congestion, divert traffic away from the incident, assist emergency vehicles and get the queued vehicles out. Coordination layer strategies are also designed for the vehicles stopped in the queue to *Backup* and then *Catch Up* with the adjacent lane traffic so as to move out.

All other strategies result in the faulty vehicle leaving the highway on its own. Note that the link layer need not be involved for these faults. For faults in the class "vehicle needs assistance to get out" a strategy called *Take Immediate Exit*(TIE) is executed by the coordination layer. The strategy consists of up to two *Forced Split* maneuvers for the faulty vehicle to become a free agent.

The free agent then executes a number of *Emergency Lane Change* maneuvers until it reaches the rightmost automated lane from where it takes the next exit. Figure 16 contains highway snapshots while the TIE maneuver is in progress.

This strategy is used by all subclasses except in cases where the vehicle capabilities limit its use. This situation is encountered if the vehicle can not sense distant objects (needed for leader operation). In this case, a modified version of TIE, called *Take Immediate Exit - Escorted* is used. The faulty vehicle leaves the highway system as part of a two vehicle platoon in which the faulty vehicle is the follower (Figure 17). This requires a *Front Dock* maneuver if the faulty vehicle is a leader of a platoon to start with. The leader of this platoon (called the *escorting vehicle*) now executes a TIE strategy until it drops off the faulty vehicle at the nearest exit.

Finally, for faults in the class "Vehicle needs no assistance to get out" a control strategy called *Take Immediate Exit - Normal* is chosen by the coordination layer supervisor. TIE-Normal is a milder version of TIE in the sense that it uses normal mode lane change maneuvers. The neighboring vehicles and the roadside are not affected by this strategy.

5.2.3 Maneuvers

To implement the above control strategies, the coordination layer supervisor makes use of the normal mode maneuvers as well as the new maneuvers, *Forced Split*, *Emergency Lane Change* and *Front Dock*. These maneuvers collectively serve as the regulator level of the extended coordination layer. In *Front Dock* (Figure 18), the last vehicle of the preceding platoon decelerates to join the faulty vehicle platoon as a leader. *Front Dock* can thus be considered as a reciprocal of the normal mode *join* maneuver. The maneuvers *Forced Split* and *Emergency Lane Change* are variations of the normal mode maneuvers *split* and *lane change*.

Due to space limitations, we do not describe these maneuvers and strategies in detail. Design and verification of the extended coordination layer controller is presented in the companion paper [14].

5.3 Regulation Layer

5.3.1 Overview

The regulation layer also consists of two levels, a supervisory level and a regulator level. The regulator consists of a set of continuous feedback controllers for each task defined by the coordination layer. The regulation layer supervisor acts as an interface between the discrete event system of the coordination layer and the continuous feedback laws of the regulation layer. We briefly describe the functional requirements of the extended regulation layer below. A detailed design of the extended regulation layer is a current research area.

5.3.2 Supervisor

The normal mode regulation supervisor of Figure 19 (originally designed in [16]), 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 supervisor 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. The normal mode regulation supervisor was designed in [16] as an *interface* between the coordination layer design of [6] and the regulation layer control laws of [10].

The above interface combines discrete event and continuous dynamical systems to form a hybrid system. Unfortunately there is no systematic way of verifying the complete hybrid system at this time. Such a proof will be required to show that the design is safe, i.e., automated vehicles do not crash with each other. Even though, the regulation layer control laws were individually proved to be locally stable ([10]) and the coordination layer was proved to be deadlock free and live ([6]), when the two were combined by using above interface and tested using the SmartPath simulator ([19]), some vehicle crashes were observed ([20]). Some of the regulation layer maneuvers were redesigned ([15]) and the safety criterion in the interface were modified ([21]) to enhance the safety of the system. The mathematical verification of *safety* of the normal mode design is still under way (see [22] for an optimal control approach to verification of this hybrid system).

The degraded mode regulation supervisor will be required to play a similar role. The outline of the finite state machine is shown in Figure 20. All the details of this design can not be completed before the feedback control laws for the new maneuvers are specified. Design and verification of an interface to guarantee safety of the vehicles in presence of faults is still an active research area.

5.3.3 Control Laws

Most of the coordination layer maneuvers described above can be carried out by tuning some of the regulation layer feedback control laws designed for normal operation. For example, the maneuvers ELC and FS will use the normal mode regulation layer lane change and split feedback control laws respectively ([10, 15]). The new maneuvers front dock and platoon lane change (needed for TIE-E and queue management) need separate regulation layer control laws to be designed.

5.4 Physical Layer

5.4.1 Normal Mode Requirements

For normal operation, each vehicle should be capable of detecting the relative distance and velocity from the car in front, in its own lane, apart from measuring its own velocity and acceleration. The necessary range of this sensing depends on the maximum speed allowed on the highway and the maximum deceleration a vehicle can apply. It takes $90m$ for a vehicle to stop from the maximum speed of $30m/sec$ using the maximum deceleration of $0.5m/s^2$, thereby requiring the sensor range to be at least $90m$. If the lane change maneuver is not restricted to take place at certain locations[5], then we also need $90m$ range for the sensors detecting relative distance and velocity of the vehicles in the adjacent lane (in both front and rear of the vehicle). Actuators for acceleration, braking and steering are needed for automatic operation of the vehicles in an AHS. The coordination layer protocols need

[5]Lane changes can alternatively be designed so as to take place only at certain fixed locations on the highway. Such an approach is used for changing lane from the transition lane to the automated lane in the entry maneuver design of [11]. In this case, the lane change is assisted by roadside sensors allowing on-board vehicle lateral sensors to have a reasonably small range.

inter-vehicle communication capability that ranges over at least the sensor range on the front and the rear of the vehicle and spanning across two lanes on either side. The follower control laws of [7] need infrared communication link to transmit the acceleration of the lead vehicle of the platoon. The normal mode architecture also requires radio communication capability between link layer and individual vehicles as well as fixed communication infrastructure for link-to-link data transmission.

5.4.2 Degraded Mode Requirements

The degraded mode architecture proposed here only keeps track of functionality or the capability of the physical layer. The capability can be enhanced by using multiple redundant sensors along with sensor fusion. Sensor fusion schemes for AHS are discussed in detail in [13]. We also assume that fault detection methods exist on the vehicle and the infrastructure which together with the sensor fusion methods, appropriately fill out the capability vector of the physical layer.

For the degraded mode architecture, we need longitudinal distance and rate sensors detecting rear vehicles, in addition to the sensors used for normal mode controllers. The number of redundant sensors for each functionality should be calculated based on the failure probability of sensors. Similar to the predicate hierarchy of Section 3.1, the failure probability of individual components will propagate into probability of crashes and injuries. The goal of the project is to design a fault tolerant architecture and controllers so as to substantially reduce probability of collisions and injuries over the current manual traffic.

6 Discussion & Further Issues

The overall fault tolerant control architecture can be described with the help of Figure 21. The system starts operation in the normal mode. There are two loops starting and ending in normal mode. The robustness analysis loop represents the on-line parameter tuning described in Section 3.2. In case of severe disturbance or loss of capability due to a fault, the bigger loop involving degraded modes of operation is invoked. Each vehicle is supposed to be equipped with fault detection mechanisms. Once a fault is detected[6], the appropriate predicate for the physical layer capability changes from 1 to 0. This change propagates upwards through the predicate hierarchy of Section 3.1. The *fault handling* module of Figure 21 is responsible for fault classification based on the state of the capability monitor. The coordination supervisor of the faulty vehicle then selects the optimal[7] strategy among the ones that are possible[8]. The selected degraded mode strategy is then carried out by the faulty vehicle. This strategy will often need assistance from the neighboring vehicles. Cooperation from the neighbors is guaranteed by assigning the degraded mode strategies higher priority than the normal mode strategy. Thus, if the faulty vehicle or the neighbors required in the degraded mode maneuver execution are already engaged in a normal mode maneuver, then the normal mode maneuver will be aborted. This scheme works well in case of isolated faults on the highway. The request for a degraded mode maneuver by the faulty vehicle can get rejected if the neighbor itself is engaged in a higher priority degraded mode maneuver. The faulty vehicle in this case chooses the next best strategy until the only possible alternative is to stop on the highway. See [14] for a detailed discussion on priorities among different strategies. The degraded mode maneuver can also get aborted by the regulation layer of the vehicles involved for safety reasons. This type of abort will be issued by the interface or the regulation supervisor if a maneuver following a particular control law becomes safety critical. The *degraded mode* state of Figure 21 represents the controllers designed for degraded modes of operation. After successful completion of the degraded mode strategy, the faulty vehicle is removed from the highway (either by a tow truck or from the assistance of the neighbors) and all other vehicles on the highway return back to normal mode of operation using the *recovery* state (Figure 21) controllers. Although we have shown recovery state explicitly in the diagram, we do not expect to have special recovery laws for the coordination and the regulation layer. Special recovery laws may be needed by the link layer to assigns optimum speeds and lane change restrictions so as to bring traffic stopped behind an accident back to the normal state.

For the duration of the fault, the faulty vehicle and the neighboring vehicles[9] operate in a degraded mode wherein the speeds and throughput considerations get lower priority than the safety of the faulty vehicle. The proposed architecture is an attempt at keeping the performance degradation to a minimum by localizing the extent of a fault.

6.1 Design Optimality

The design of controllers for degraded modes involves tradeoffs between safety, complexity of coordination and control, and performance degradation of the AHS. Performance degradation is measured in terms of congestion (loss of throughput) and discomfort to the passengers caused by the malfunction. In the control design, topmost priority should be given to the safety of the vehicles. There can be many designs satisfying a given safety requirement which differ in the other two criteria, namely,

[6] See [13] for fault detection and sensor fusion details

[7] Refer to [14] for the priorities assigned to different strategies.

[8] Note that *normal mode* strategy will no longer be possible due to the fault, unless the fault belongs to the category "vehicle does not need to get out".

[9] The size of the neighborhood depends on the type of fault

control complexity and performance degradation. For example consider the following solution to the degraded mode control design problem:

> A faulty vehicle always stops on the highway regardless of the type of fault. In case of faults that limit the capability of the vehicle to perform basic driving functions, the stopped vehicle waits for an emergency vehicle to tow it out of the highway. Otherwise, traffic in all other lanes towards the direction of the exit is stopped for the faulty vehicle to exit by itself.

This strategy involves minimal coordination between vehicles as the faulty vehicle can stop by itself in most cases. The roadside controller then needs to stop the traffic in the other lanes to let this vehicle exit the highway. But a stopped vehicle on the highway results in severe loss of throughput and can create massive congestion depending on the traffic density. By using this strategy, we can maintain safety and simplicity at the cost of throughput. Moreover we need to make extensive use of the roadside controller in a safety critical way.

Our approach has been to design control laws to achieve the objective of taking the faulty vehicle to the nearest exit without stopping it on the highway[10]. As already discussed, the lower layers have access to more detailed information and operate at a faster time scale. They are therefore better suited to assess the safety of a given situation. We have tried to delegate as many decisions and control actions as possible to the lower layers of the control hierarchy in order to make the system more robust to error. This localization of failures has been our first criterion for optimality of the extended architecture. As a result, our control laws need assistance from the neighboring vehicles thereby increasing complexity. Overall, our design maintains safety and reduces performance degradation at the cost of complexity of control.

It should be noted that, because of its complexity, formulation and solution of the degraded modes problem in terms of optimal control is very difficult. The fact that we came up with a hierarchical design and based it on a classification of faults implies that our solution will probably not be optimal in any meaningful cost criterion. Our design represents a particular compromise between safety[11], capacity and complexity. It is yet to be verified that our approach results in less performance degradation than the simple strategy described above. This is a topic of further research.

6.2 Verification Issues

After designing the control laws in this framework, the extended architecture has to be verified before implementation. The extended coordination layer control laws have been verified to be deadlock free, live and fair [14]. The above discrete proof makes simple abstractions of the continuous dynamics of the regulation and physical layer such as; the regulation layer always follows the orders of the coordination layer, it completes each maneuver without colliding with other cars, will not abort a maneuver infinitely often and will be collision free even after aborting a maneuver. This proof will suffice to guarantee safety of vehicles if the continuous dynamics can be designed to obey the abstractions. From past experience [10, 16, 20] it is clear that the regulation layer control laws can not be designed to satisfy such abstractions in presence of a wide variety of disturbances. The difficulty

[10] In certain cases such as faults in Appendix A.1, stopping the vehicle is the only alternative for safety. To improve the system performance the hardware design must keep the probability of occurrence of these faults to a minimum.

[11] It is an ongoing effort to prove that the extended architecture is in fact safe. Proving system safety will involve the hybrid system resulting from the interactions of the coordination and the regulation layer. From our past experience, it seems plausible that for our framework (and the coordination layer of [14]), suitable regulation layer controllers can be designed to satisfy the safety requirement.

arises because of the actuator constraints, sensor limitations and constraints on the potential of the vehicle [20]. One can design locally stable regulation layer control laws for each maneuver. The problem now reduces to showing that for all possible state trajectories (traces) of the combined hybrid system (coordination plus the regulation and physical layer), the continuous state of each vehicle at every instant of time is in the region of attraction (safe set) of the regulation layer control law that the vehicle is applying. Such a hybrid system design would require a method to calculate the reachability sets of continuous dynamical systems. Unfortunately no such method exists in the literature. Consequently, there are no tools to verify and design hybrid systems at the moment.

Because of this lack of tools, simulation plays a very important (if not indispensable) role in the design of complex, hybrid systems. Even though simulation can not replace formal proof techniques (analytical or computational) it can still provide valuable information about the system performance. More specifically, successful results under extensive simulation indicate that the design is likely to behave well, even though, usually there is still a lot of room left for situations where the system behaves poorly. On the other hand, unsatisfactory performance on the simulation testbed indicates that the design is not good enough for certain cases and may suggest improvements that will eliminate these shortcomings. In other words, simulation results can not be taken as proof that a system works well in general but they can be taken as proof that it works in specific cases, or, more importantly, that it doesn't work in others. The AHS simulator *SmartPath* [19] has been used successfully in the past to identify shortcomings of the hybrid system and improve the design. In [20], it was shown that even for the normal mode design, separate proofs of coordination and regulation layer behavior are not sufficient to prove safety of the combined system.

We are approaching this issue in the following ways.

- We are extending the capabilities of *SmartPath* to include working under degraded modes of operations. The controllers for extended architecture will be implemented in *SmartPath* along with the ability to induce faults and adverse environmental conditions. We hope to identify some of the system shortcomings by simulation.

- Following the approach suggested in [22], we are developing optimal control and game theoretical methods to analytically verify safety and reachability properties of the hybrid dynamical system of PATH.

- We are also working on extending the framework mentioned in this paper to handle probabilistic data. Then one would be able to calculate probability of a crash given probabilities of failures of different subsystems.

After proving the safety of the extended architecture, we need to estimate the performance (throughput, time delays) in the presence of faults. This will allow one to compare different fault tolerant designs for optimality. Two different methods can be employed to achieve this. Firstly, Monte Carlo simulations can be performed using SmartPath to obtain performance metrics. Alternatively, an aggregate traffic flow model can be developed to predict the traffic flow under degraded modes of operation. The flow model can be validated using SmartPath simulations. Research efforts in developing an extended flow model and using it to design extended link layer supervisor controller are currently under way.

7 Concluding Remarks & Future Work Directions

We proposed a framework for an AHS design that is capable of operating in the presence of faults and other factors that induce performance degradation (such as adverse weather conditions). Our framework is hierarchical and builds on the control architecture of [4]. The design provides a high degree of autonomy by extending the information structure to include data about the system capability and the control structure to make a distinction between strategic planning and execution.

Our framework now needs to be filled in with appropriate control laws. We proposed a design for the coordination layer and gave requirements for the link and regulation layer controllers.The assumption our design makes about the physical layer were also stated. The complete coordination layer design, together with verification results is given in the companion paper [14]. Work is already underway for a link layer design compatible with our framework, both for the planning [17] and the regulator [18] levels. The modifications to the regulation layer design are the subject of ongoing research.

Our design raises important issues for both automated highway systems and hierarchical control in general. Ideally we would like to produce an optimal compromise between the complexity of the design, highway throughput, passenger comfort and safety. Unfortunately such an optimum is very difficult to obtain, due to the complexity of the problem. In fact at the current stage there is no formal technique for determining the safety and performance of a proposed design. We tried to touch upon these issues in Section 6. We are currently working on resolving some of these problems and providing a framework for evaluating the performance of an AHS design in terms of capacity and safety.

Acknowledgment: The authors would like to thank Luis Alvarez, Akash Deshpande, Jonathan Frankel, Roberto Horowitz, Perry Li, Antonia Lindsey, Anuj Puri, Shankar Sastry, Ekta Singh and Pravin Varaiya for helpful discussions providing insight into this problem. We are also grateful to Bret Foreman, Chris Gerdes, Dragod Maciuca, Dieter Koller, Satyajit Patwardhan and D. Swaroop for their help in obtaining the information contained in Appendix A and B

References

A List of Faults

All faults are listed in their corresponding classes and subclasses, according to specification of Section 4.

A.1 Vehicle stopped/must stop

1. no throttle control, no engine power, out of gas, no power transfer, vehicle to vehicle communication down (Radio - Needed for coordination)
2. no steering control, uncontrolled object ahead, no control computer, magnetometer failure, no sensing of distance and velocity of car ahead (long and short range)
3. no brake control

A.2 Vehicle needs assistance to get out

1. no control of transmission / selection of gear
2. no long-range (longitudinal) sensing of vehicles
3. no short-range (longitudinal) sensing of vehicles
4. no lateral sensing of vehicles
5. flat tire - reduced steering capability
6. Vehicle - Vehicle communication down (Infra-Red: Needed for Follower operation)

A.3 Vehicle needs no assistance to get out

1. Non-crucial Sensor fault: engine sensor (e.g. intake manifold pressure sensor), accelerometer, wheel speed, etc.
2. low on gas
3. Single fault in a redundant sensor set
4. Vehicle-roadside communication down because of on-board equipment failure

A.4 Vehicle does not need to get out

1. Lights won't go on
2. In vehicle displays not working
3. Out of range of magnets, magnetometers working, not changing lanes

A.5 Infrastructure Failures

1. Network layer down or communication between link and network down
2. Link layer down or communication between link and vehicle down in the entire link due to roadside equipment failure
3. unable to communicate with object on AHS
4. Lane(s) Blocked
5. Exit(s) Closed
6. Entry(s) Closed
7. Robustness Spill Over and environment
   In this category we group all problems caused by unfavorable conditions that the normal mode controller is not robust enough to handle. These problems may not be the result of faults, but may arise due to gradual performance degradation. Since they result in certain normal mode predicates calculating as false, however, degraded modes will have to be designed for them. If the gradual performance degradation is limited to a single vehicle then it will be classified into one of the classes 3.3.1 through 3.3.4. Here we consider the effect on the infrastructure. This is mainly caused due to environmental degradation such as rain or snow. They will be grouped in two subclasses:
   - loss or reduced traction with road (lateral & longitudinal)
   - reduction in sensor range or accuracy (caused by rain, dust, sunshine, etc.)

A.6 Driver/Computer Interaction Down

Problems in this class mainly occur during the entry and exit to the system. We assume that once on the freeway the driver may not interfere with the system operation and therefore can not induce any special faults.

1. Improper Exit: Driver unable to take control and/or system unable to transfer control at exit
2. Improper Entry: Wrong destination/route entered by driver or system unable to start automatic control at entrance or manual driver tries to enter automated TL

A.7 Faults not considered

1. Software implementation errors
2. Design errors such as protocol design errors, control design errors
3. Communications errors including: wrong message, message to wrong car, etc.

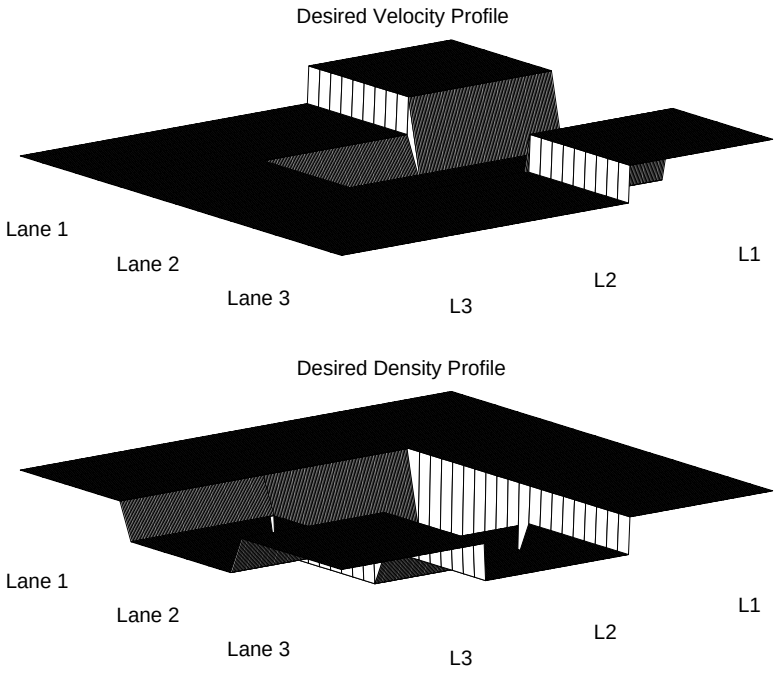

Figure 15: Desired velocity & density profiles for Stage 2

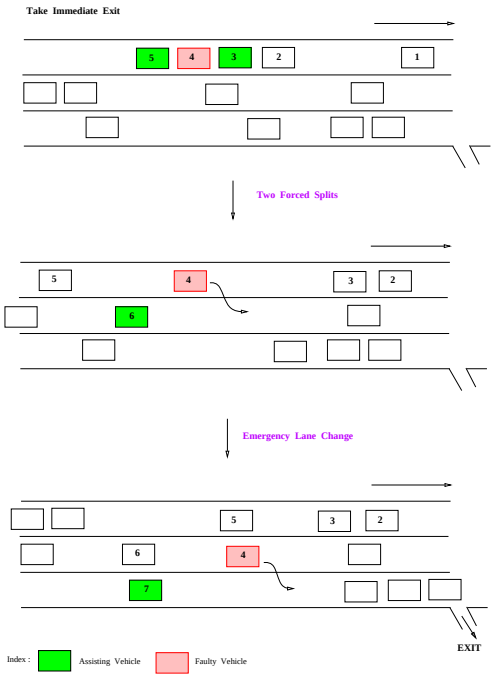

Figure 16: Take Immediate Exit: Highway Snapshots

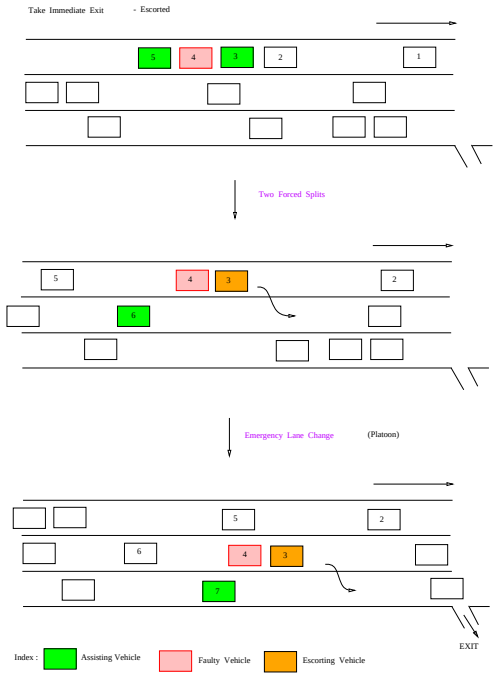

Figure 17: Take Immediate Exit - Escorted: Highway Snapshots

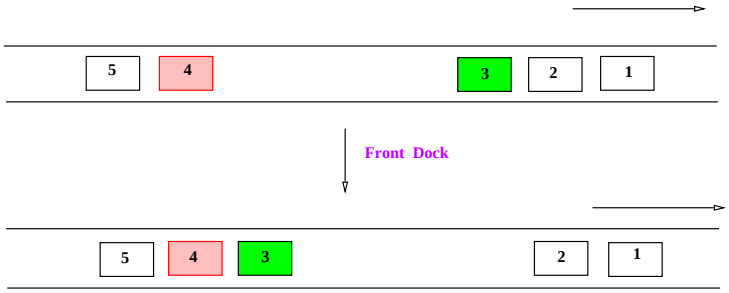

Figure 18: Front Dock Maneuver

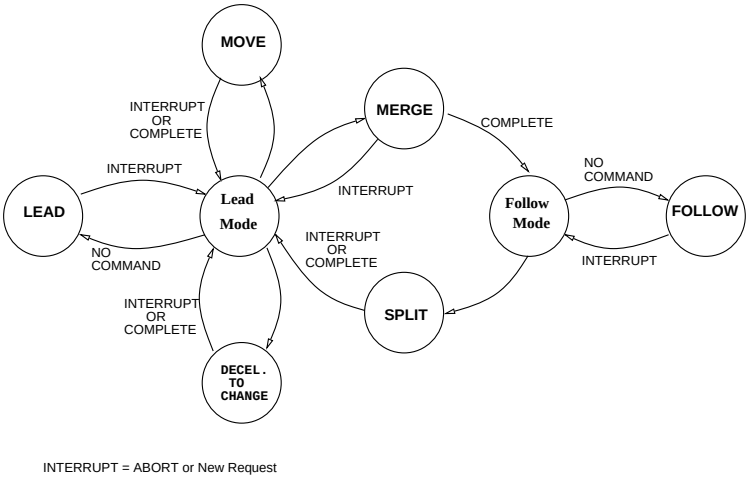

Figure 19: Normal mode regulation layer supervisor

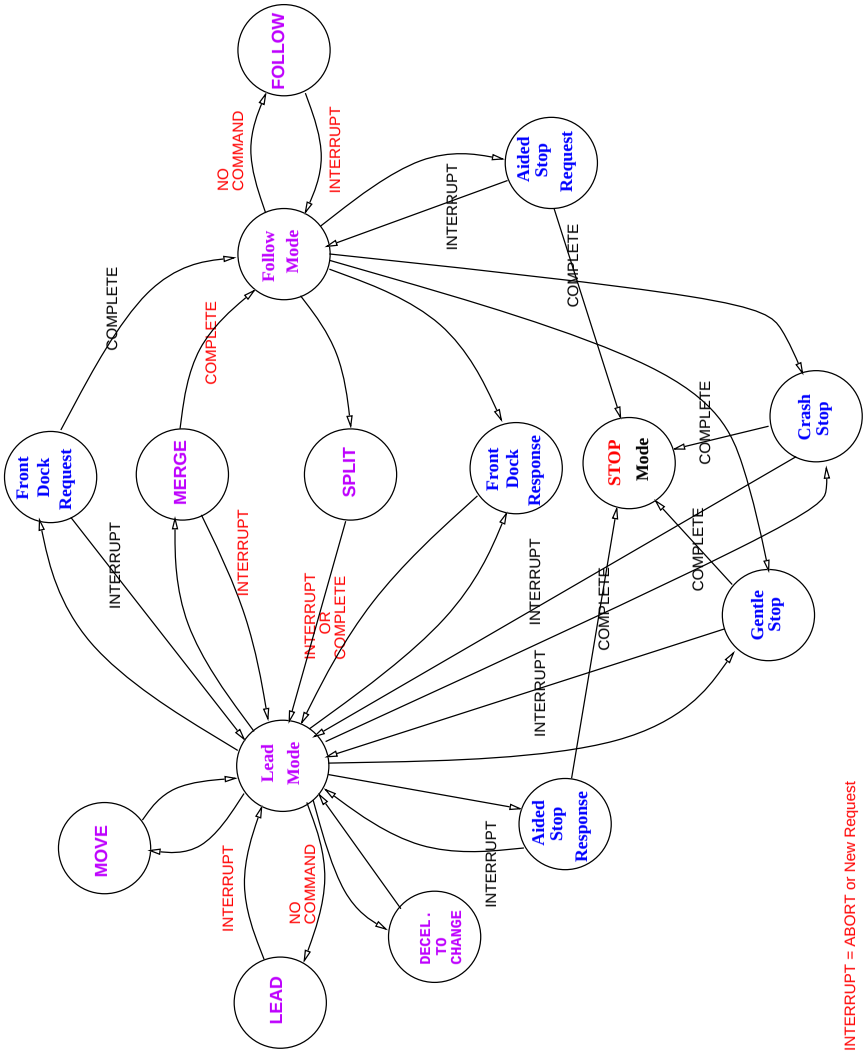

Figure 20: Outline of the degraded mode regulation layer supervisor

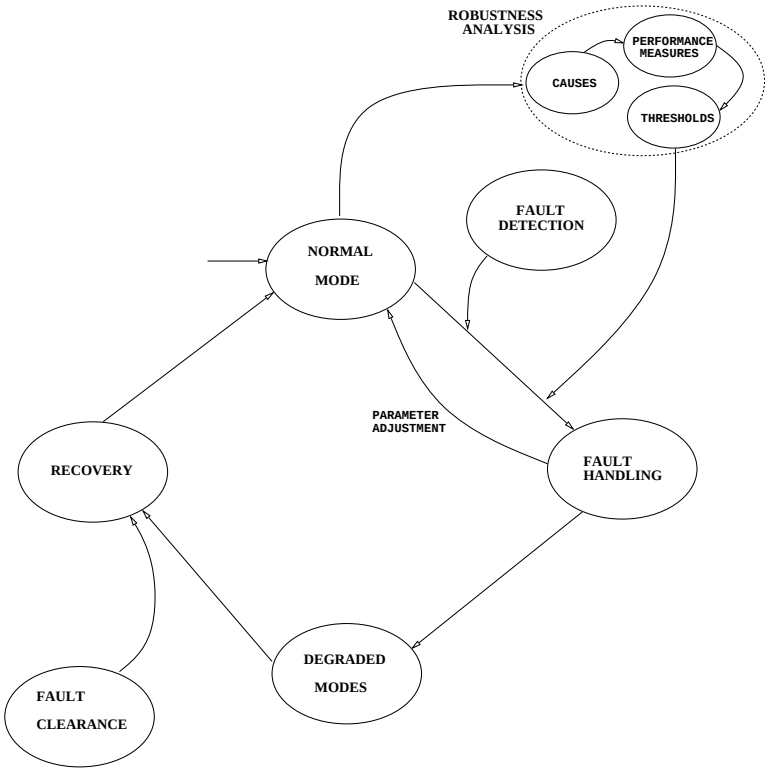

Figure 21: Fault Tolerant architecture: Fault handing and Robustness enhancement

B Causes of Gradual Performance Degradation

B.1 List of Causes:

Environment

1. Longitudinal Wind
2. Lateral Wind
3. Sunshine
4. Sunrise-Sunset
5. Ice
6. Snow
7. Fog
8. Rain
9. Wet Road
10. Oil on Road
11. Pot Holes
12. Gravel
13. Other Debris
14. Road Slope
15. Road Bank
16. Road Curvature
17. Slip Stream of Other Vehicles
18. Magnet Damage
19. Lane Marker Damage

Vehicle

20. Vehicle Make
21. Tire Pressure
22. Brake Fluid Pressure
23. Suspension
24. Longitudinal Sensors
25. Lateral Sensors
26. Radio Link
27. Radar
28. Accelerometer
29. Cameras
30. Steering Error
31. Accelerator Error
32. Braking Error
33. Brake Fade
34. Brake Wear
35. Yaw Rate Sensor

B.2 Performance Parameters

$$\mathcal{P} = P_P \cup P_S \cup P_R \cup P_C \cup P_L \cup P_N$$

Physical

- P_P^1: Maximum Acceleration
- P_P^2: Minimum Acceleration
- P_P^3: Maximum Velocity
- P_P^4: Maximum Cornering Stiffness

Sensors & Communications

- P_S^1: Maximum Sensor Range
- P_S^2: Measurements Mean and Variance
- P_S^3: Lost Packets

Regulation

- P_R^1: Maximum Longitudinal Tracking Error
- P_R^2: Maximum Lateral Tracking Error

Coordination

- P_C^1: % of maneuvers that get aborted
- P_C^2: Timers that expire

Link

- P_L^1: Density error
- P_L^2: Flow error
- P_L^3: % of vehicles that miss exit

Note: In general performance parameters of higher levels depend on those of lower levels.

B.3 Qualitative Dependencies

In this section, we determine the qualitative dependencied between the causes of performance degradation and the performance parameters described in the previous sections. The detailed functional form can be calculated only after conducting experiments with the actual hardware and the specific control laws in the environmental conditions mentioned above.

B.3.1 Physical Layer

Maximum Acceleration

$P_P^1 = f^i(C_{14})$

- $P_P^1 \approx 2m/s^2$ on flat road
- $P_P^1 \propto \sin(C_{14})$

Minimum Acceleration

$P_P^2 = f^i(C_5, C_6, C_9, C_{14}, C_{10}, C_{12}, C_{21}, C_{22}, C_{23}, C_{32}, C_{33}, C_{34})$

- Assuming C_5, C_9, C_{10} & C_{12} are binary:

$$P_P^2 \approx \begin{cases} 0.2g & \text{if } C_5 \bigvee C_6 \bigvee C_9 \bigvee C_{10} \bigvee C_{12} \\ 0.7 - 1g & \text{otherwise} \end{cases} \quad (1)$$

- $P_P^2 \propto \sin(C_{14})$
- Effect of C_{22} is approximately linear
- C_{33} and C_{34} will have minimal effect on automated highway
- Absence of low level controllers makes C_{32} relevant.
- Associated lag $\sim$ 50ms

Maximum Velocity

$P_P^3 = f^i(C_5, C_6, C_8, C_9, C_{10}, C_{12}, C_{14}, C_{21})$

Cornering Stiffness

$P_P^4 = f^i(C_5, C_6, C_8, C_9, C_{10}, C_{12})$

B.3.2 Sensors and Communications

Vision System

1. $P_{B.2} = f_{B.2}^i(E1, E2, E3, E4, E11, E12, E13, E14, V23, V29)$
2. $P_{B.2} = f_{B.2}^i(E1, E2, E3, E4, E5, E6, E9, E11, E12, E13, E15, E19, V23, V29)$
   - Effect of E1, E2, E11, E12, E13, V23 is similar. They produce *camera shake* which induces error in lateral as well as longitudinal sensing.
   - E5, E6, E9 make detection of lane marker difficult, thereby producing the same effect as E19 on lateral sensing error.
   - Effect of V29 : The camera might need recalibration while on the freeway thereby causing an error if there is no on-line calibration.
   - E3 and E4 produce sensing errors because of the shadows and highlights.

B.3.3 Regulation Layer

Maximum Longitudinal Tracking Error

$P_R^1 = f^i(P_P^1, P_P^2, P_S^1, P_S^2, P_S^3, C_1, C_{14}, C_{17}, C_{24}, C_{26}, C_{27}, C_{28}, C_{31})$

- Effects of C_1, C_{14} & C_{17} are similar to one another. We can expect $|P_R^1| \approx 12$cm.
- Effect of C_{24}, C_{26} & C_{27} is mainly through lag (~ 75ms).
- Low level controllers guarantee effect of C_{31} is minimal. Associated lag ~ 7.5ms, negligible.
- Effect of C_{28} is because of noisy measurements, offsets & lag.

Lateral Tracking Error

$P_R^2 = f_i(P_P^3, P_P^4, C_2, C_{15}, C_{16}, C_{18}, C_{21}, C_{11}, C_{13}, C_{23}, C_{30})$

- Effect of C_{21} is same as P_P^4 if all tires have low pressure.
- The effect of C_{21} with imbalance of tire pressures, C_{11}, C_{13}, C_{23} is similar. It is qualitatively same as a tire burst. For tire burst, $|P_R^2| \approx 10$cm on a straight road and 15-20 cm during a curve.
- Effect of C_2, C_{17} is similar. Correct road banking (C_{15}) is helpful during a curvature. The effect of incorrect banking angle is same as C_2, C_{17}.
- Effect of C_{18} is minimal as the controller can tolerate up to 5 magnets damaged in a row on a straight road.
- Presence of low level controller makes effect of C_{30} negligible. The steering actuator has a 0.1 sec. time lag.
- Effect of C_{25}, C_{28}, C_{35} is under investigation.

B.3.4 Co-ordination Layer

1. $P_C^1 = f^i(P_R^1, P_R^2, P_S^2)$
2. $P_C^2 = f^i(P_R^1, P_R^2, P_S^2, P_S^1)$

B.3.5 Link Layer

1. $P_L^1 = f^i(P_C^1, P_R^1, P_R^2)$
2. $P_L^2 = f^i(P_C^1, P_R^1, P_R^2)$
3. $P_L^3 = f^i(P_C^1, P_R^1, P_R^2)$

C Capability Predicate Hierarchy for Degraded Mode Control Strategies