Title
Fault Diagnosis for Intra-platoon Communications

Authors
Simsek, Hidayet Tunc
Sengupta, Raja
Yovine, Sergio
et al.

Publication Date
1999-07-01

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

Fault Diagnosis for Intra-platoon Communications

**Hidayet Tunc Simsek, Raja Sengupta,
Sergio Yovine, Farokh Eskafi**

California PATH Research Report
UCB-ITS-PRR-99-24

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

Report for MOU 332

July 1999

ISSN 1055-1425

Fault Diagnosis for Intra-platoon Communications (MOU332 Final Report)

Hidayet Tunç Şimşek, Raja Sengupta, Sergio Yovine, Farokh Eskafi
{simsek,raja,sergio,farokh}@path.berkeley.edu

California PATH, UC Berkeley
Richmond Field Station Bldg. 452
1301 S. 46th St, Richmond CA 94804

February 26, 1999

1 Introduction

We are interested in studying the fault diagnostics of platooning vehicles. It is understood that a platoon is a string of vehicles with distributed control strategies. Vehicles rely on real-time control data from other vehicles for correct execution of their control laws. A time-driven system is responsible for delivering the control data.

This document formalizes the design and logical verification of a fault diagnosis and monitoring system for intra-platoon communication systems. The design presented here is motivated by the ideas given in [1, 2, 3, 4]. The design is formally specified using the SHIFT programming language, [5, 6, 7, 8], for networks of hybrid automata. We embellish the SHIFT specification with the meta language GRIZZLY, [9]. Finally, the SHIFT-KRONOS connection, [9, 10], is used to verify the logical correctness of the design. One advantage of this approach is that the same specification is used for both verification and simulation. This aspect is emphasized since verification of the specification refers to the correctness of the underlying logic whereas simulation deals with performance issues. In addition, ongoing research at PATH will allow us to generate real-time code suitable for implementation from the same specification.

The next section summarizes the system requirements for a category of intra-platoon communication systems studied at California PATH. By doing so we try to clarify and state precisely the diagnosis problem that is the subject of the remainder of this report. In Section 3 we revise our mathematical model for general diagnoser systems. In Section 4 we present in detail the complete design of our diagnoser system for intra-platoon communication systems. Section 6

Radio Device	*Modulation*
WaveLAN	Direct sequence spread spectrum
Infra-red	Clock encoding
Broadcast	Frequency hopping spread spectrum modulation
Broadcast	CDMA spread spectrum modulation

Table 1: Intra-platoon radio devices.

presents the computer aided verification of our design. Finally, in Section 7 we summarize the verification results.

2 Intra-platoon communication systems

Four kinds of radio devices have been considered at PATH for intra-platoon communications. We do not deal with the analysis of these device configurations, however, we have listed them in Table 1 to illustrate the domain of our fault diagnosis problem. We have learned from experience that these configurations are representative of a large class of intra-platoon communication systems and as such we will formally specify the diagnoser problem within their context. For a detailed analysis of radio devices refer to [11, 12, 13, 14].

Figure 1 illustrates the overall layout of the intra-platoon communication networks. The communication domain consists of a wide area network (WAN) and several local area networks (LAN). A message is called *non-critical* if there is no timing constraint associated with it, and similarly a message is called *critical* if its content expires with time. Vehicles may transmit non-critical messages over the WAN to another vehicle (point-to-point), group of vehicles (multi-cast) or the the entire network (broadcast). The WAN is a fixed and reliable[1]. The WAN consists of a dedicated radio in each vehicle, and the fixed access points (FAP) on the road-side. Vehicles transmit critical messages[2] over the LAN. A LAN is a temporary network and unreliable network that is created within platoons and message passing over the LAN is restricted to that platoon. The LAN consists of a dedicated radio in each vehicle.
Messages transmitted over the LAN contain control data such as vehicle velocity, acceleration, etc. A typical LAN communication scheme has the following properties (refer to Figure 2):

- Each vehicle receives a periodic update message (control message) from the lead vehicle and vehicle-in-front,
- Each vehicle transmits a periodic control message to the vehicle-in-back every 20ms. (The lead vehicle transmits to every vehicle in the platoon),

[1]Messages eventually reach their destinations.
[2]These critical messages are assumed to be the real-time control data

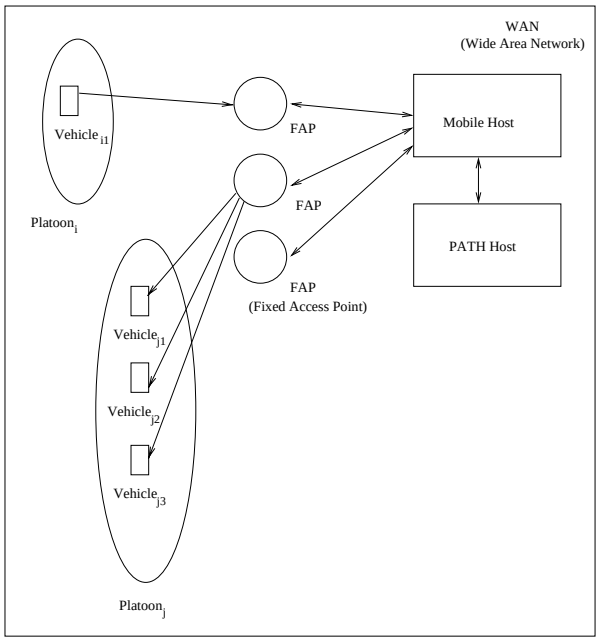

Figure 1: Overall layout of radio networks.

- The lead vehicle does not receive any messages from the vehicles in its platoon over the LAN,
- The last vehicle does not transmit any messages over the LAN.

Messages transmitted over the WAN contain regulation data. Such data is used for regulated maneuvers; e.g. lane change, merging to a platoon, splitting from a platoon, etc.

2.1 Formal statement of the diagnoser design problem

We assume a layered architecture for the LAN. Our purpose is to design a diagnostic system that resides at the application level of this hierarchy. The proposed hierarchy is illustrated in Figure 3.

3 Mathematical Background

This section reviews the notions of a plant, a decentralized diagnoser system and the results obtained thereof from [1]. The formalism adopted here is that of discrete event systems, [1, 15, 16, 17]. The reader may also be interested in studying the Petri net approach for modeling fault diagnosis in distributed systems, [18, 19].

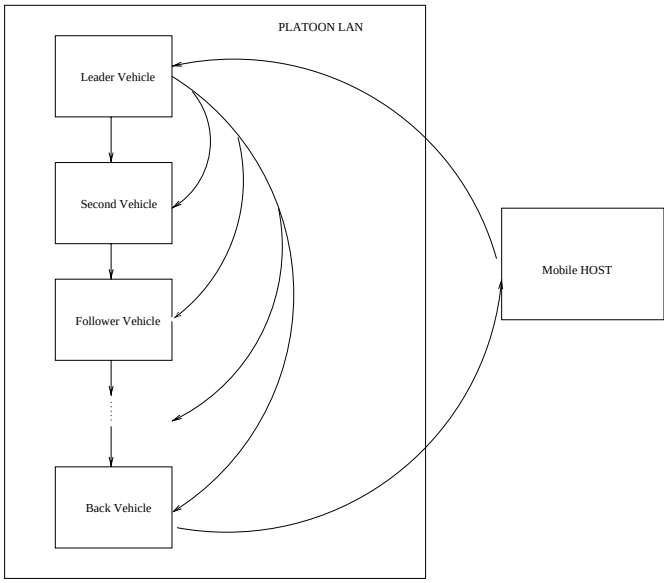

Figure 2: Message passing within a platoon.

3.1 Decentralized diagnoser system architecture

The decentralized diagnoser system architecture of Figure 4 illustrates the adopted architecture for our design. The system for which the diagnosers are to be designed is called the *plant*, P. The plant is modeled as a language $L_p \subseteq \Sigma_p^*$ over the finite alphabet Σ_p. The plant may concurrently and asynchronously execute multiple processes where the behavior of a process is represented by a string $w \in L_p$. Concurrency is modeled with interleaved semantics. An event $\sigma_p \in \Sigma_p$ is said to be observable to a diagnoser d_i when d_i is capable of recording the occurrence of that event in a string generated by the plant. Observable plant events are denoted by $\Sigma_o \subseteq \Sigma_p$ and unobservable plant events by $\Sigma_{uo} \subseteq \Sigma_p$ and it should be intuitively clear that $\Sigma_p = \Sigma_o \uplus \Sigma_{uo}$[3]. Of significant importance is a subset, Σ_f, of the unobservable events that represents the faults in the plant: $\Sigma_f \subseteq \Sigma_{uo} \subseteq \Sigma_p$.

Assume that there are n diagnosers. Each diagnoser, d_i, is capable of observing a subset, Σ_{d_i}, of the observable events generated by the plant: $\Sigma_{d_i} \subseteq \Sigma_o \subseteq \Sigma_p$. Each diagnoser, d_i may generate and communicate estimates (messages), $\sigma_{m_i} \in \Sigma_{m_i}$ to any other diagnoser via a reliable communication channel that is assumed to be different from any communication channel used by the plant. This communication of estimates is modeled with the function $\nu_i : \Sigma_{d_i}^* \rightarrow \Sigma_{m_i}$.

We will associate with each diagnoser, d_i, a subset of the plant failure events

[3]The notation $Q = Q_1 \uplus Q_2$ is shorthand for $Q = Q_1 \cup Q_2$ and $Q_1 \cap Q_2 = \emptyset$.

Application Layer *A diagnostic protocol that can separate channel faults from hardware faults and isolate the vehicle in which the hardware fault is located. A persistent channel fault may be diagnosed as a hardware fault.*
Network Layer Assume there is a redundant WAN. ARP is assumed to be correct and there is trivial routing
Data Link Layer No LLC is required
MAC Layer Token Ring
Physical Layer Spread Spectrum/WaveLAN. Performs synchronization and CRC

Figure 3: Formal statement of the diagnoser design problem.

$\Sigma_{f_i} \subseteq \Sigma_f$ such that $\Sigma_f = \bigcup_i \Sigma_{f_i}$[4]. The diagnoser d_i is responsible for inferring whether a $\sigma_f \in \Sigma_{f_i}$ occurred. That d_i indeed inferred the occurrence of σ_f is modeled with the function $\theta_i : \Sigma_{d_i}^* \times \Sigma_m^* \rightarrow \Sigma_{mf_i}$ where $\Sigma_m = \bigcup_i \Sigma_{m_i}$. We note that for practical purposes $\Sigma_{mf_i} = \Sigma_{mf}$ for each i.

Finally, we may formally define a design.

Definition 1 (Design, L) *Let a diagnoser d_i be represented by the tuple $d_i = (\Sigma_{d_i}, \Sigma_{m_i}, \Sigma_{mf_i}, \nu_i, \theta_i)$. Then, given a plant $P = (L_p \subseteq \Sigma_p^*, \Sigma_o, \Sigma_f)$, a design is a prefix-closed language $L \subseteq \Sigma^*$ over the finite alphabet Σ with respect to the diagnosers d_i, $i = 1, \ldots, n$ where:*

- $\Sigma = \Sigma_p \uplus \Sigma_m \uplus \Sigma_{mf}$,
- $\Sigma_m = \bigcup_i \Sigma_{m_i}$, $i = 1, \ldots, n$ *is the finite estimate set of the n diagnosers,*
- $\Sigma_{mf} = \bigcup_i \Sigma_{mf_i}$, $i = 1, \ldots, n$ *is the finite fault messages of the n diagnosers.*

Intuitively, a design is a language L that captures all the interleaved behaviors of the plant, the messages transmitted amongst the diagnosers and the fault messages output by the diagnosers.

Before we can define diagnosability we will require that a design is admissible and correct.

3.2 Admissibility, Correctness and Diagnosability

The notion of an admissible design stems from the intuitive fact that the diagnosers do not force, disable plant processes, or more simply put, the diagnosers do not control the plant.

[4]Note that for $i \neq j$, $\Sigma_{f_i} \cap \Sigma_{f_j}$ is not necessarily $\emptyset$. That is, more than one diagnoser may try to isolate the same fault.

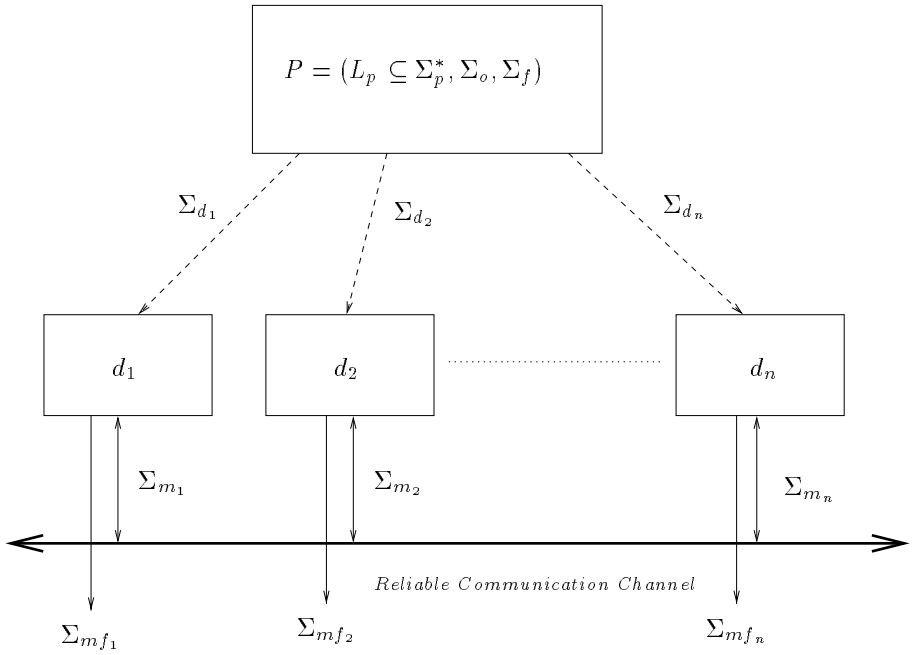

Figure 4: Decentralized Diagnoser System Architecture.

Definition 2 (Admissibility) *A design L is said to be admissible whenever:*

- *Consistency:* $L_p \subseteq L$,
- *Passivity:* $P_{\Sigma_p}(L) = L_p$,
- *Causality: For each* d_i, $\sigma_{m_i} \in \Sigma_{m_i} \cup \Sigma_{mf_i}$, $u, v \in L$: $P_{\Sigma_{d_i}}(u) = P_{\Sigma_{d_i}}(v) \Rightarrow \sigma_{m_i} \in u$ *iff* $\sigma_{m_i} \in v$.

Proposition 1 *If for each diagnoser d_i the functions ν_i, θ_i are injective then the causality condition is satisfied.*

Definition 3 (Correctness) *Assume that there exists a bijection enumerate* : $\Sigma_f \rightarrow \Sigma_{mf}$. *A design L is said to be correct if for each diagnoser i, $\sigma_f \in \Sigma_{f_i}$ occurs if and only if enumerate$(\sigma_f) \in \Sigma_{mf_i}$ occurs (with finite delay).*

Diagnosability naturally follows from the definitions of admissibility and correctness.

Definition 4 (Diagnosability) *Let $P = (L_p \subseteq \Sigma_p^*, \Sigma_o, \Sigma_f)$ be a plant. Then, P is said to be diagnosable by a design L if and only if L is admissible and correct. Furthermore, P is said to be:*

1. *Centrally diagnosable by L if L is designed with respect to a single diagnoser, d,*

2. *Decentrally diagnosable by L if $\Sigma_m \neq \emptyset$.*

Note that 2 $\Rightarrow$ 1.

3.3 Results

We will repeat here a theorem that provides a sufficient and necessary condition on the existence of a correct design.

Theorem 1 (Existence of a correct design) *Let P be a plant $P = (L_p \subseteq \Sigma_p^*, \Sigma_o, \Sigma_f)$. For P a correct design L exists $\Leftrightarrow$*

$$\begin{aligned}\exists m .[&\forall \sigma_f \in \Sigma_f,\ u, v, w \in L_p \\ &|v| > m \wedge (P_{\Sigma_{d_i}}(u\sigma_f v) = P_{\Sigma_{d_i}}(w))_{i=1,\ldots,n} \Rightarrow \sigma_f \in w]\end{aligned}$$

For a detailed discussion and proof of Theorem 1 refer to [1].

Let us concentrate on the sufficiency of the condition of Theorem 1. Note that this is a property of the plant only and does not explicitly restrict the diagnoser functions ν_i and θ_i, $i = 1, \ldots, n$.

However, in [1], the sufficiency of Theorem 1 is proven constructively and it is shown that if there exists a correct decentralized design then the *communicate all observations* design (i.e. $\nu_i : \Sigma_{d_i} \twoheadrightarrow \Sigma_{m_i}$ is a bijection) is a correct design. Conversely, if the *communicate all observations* is not a correct decentralized design then there exists no correct decentralized design.

4 Formal description of plant and diagnosers

We will describe the plant, communication channel and diagnosers formally. For this purpose we will use SHIFT, a hybrid system programming language with simulation semantics, and KRONOS, a timed automata verification tool. The gap between simulation and verification is bridged through the meta-language GRIZZLY, an extension of the SHIFT programming language. We would like to maintain the notions adopted in Section 3, which means that we must use a discrete event formalism with interleaved semantics to represent concurrency. We will use the discrete event formalism of SHIFT to model the system and we will embellish this model with real-time constraints using GRIZZLY.

4.1 Channel Model

We will maintain that the channel characteristics varies significantly from vehicle-to-vehicle. For a detailed discussion of channel characteristics and channel parameters refer to [11]. We will categorize the communication channels to emphasize that these channels may implicitly have different characteristics:

Synchronization The synchronization channel carries the control packet, sm, transmitted by the lead vehicle, l. This is a single-node broadcast channel where all vehicles except the leader are recipients of this message. The synchronization message, sm, is a control packet with a synchronization prefix that the recipient vehicles use to synchronize their clocks with the leaders. Vehicles with unsynchronized clocks cannot transmit or receive messages.

Control The control message channel represents a channel between two consecutive vehicles. It carries the control message, cm, from the vehicle-in-front to the vehicle-in-back.

Loopback The loopback message channel represents a channel between two consecutive vehicles. The loopback message, lbm, is carried from the vehicle-in-back to the vehicle-in-front.

We will implement all three types of channels using a common timed automaton. Figure 5 illustrates the channel model. We will use the shorthand $Channel^{m}_{Xmitter \rightarrow Receiver}$ to denote a uni-directional channel from $Xmitter$ to $Receiver$. The superscript m denotes the channel type and $m \in \{sm, cm, lbm\}$. We will soon see that $Xmitter$, $Receiver \in \{l, s, f, b\}$ where l stands for *lead* vehicle, s for *second* vehicle, f for *follower* vehicle and b for *back* vehicle.

Normally, the channel remains in the $getMessage$ mode until it is interrupted by some transmitter, $Xmitter$, requesting a message, msg, to be sent to some receiver, $Receiver$. In that case, the channel moves to the $putMessage$ mode and thereafter determines whether that message should successfully go through. The boolean function f should be a statistical model representing the transmission characteristics of the channel; specifically, f will be different for synchronization, control and loopback channels. Finally, the channel model returns to the $getMessage$ mode.

The auxiliary variable msg represents the message of the transmitter, $Xmitter$. Its value is irrelevant for verification since we are not concerned with the contents of the control packets. The auxiliary variable rcv is used to select the target receiver when f is $true$. rcv is nil if f is $false$.

Note that this channel is uni-directional and that the receiver, $Receiver$, is expected to have a non-blocking transition that expects to synchronize on $m \in \{sm, cm, lbm\}$.

4.2 Plant model

In this section we will model the plant (i.e. the LAN of a platoon given in Figure 2.) We will classify the vehicles in a platoon into 4 different categories as illustrated in Figure 6:

Lead Vehicle, l The lead vehicle differs from other vehicles in that it does not receive any messages from any other vehicles. Furthermore, its control message is prefixed by a synchronization pattern, hence we call it the synchronization message, sm.

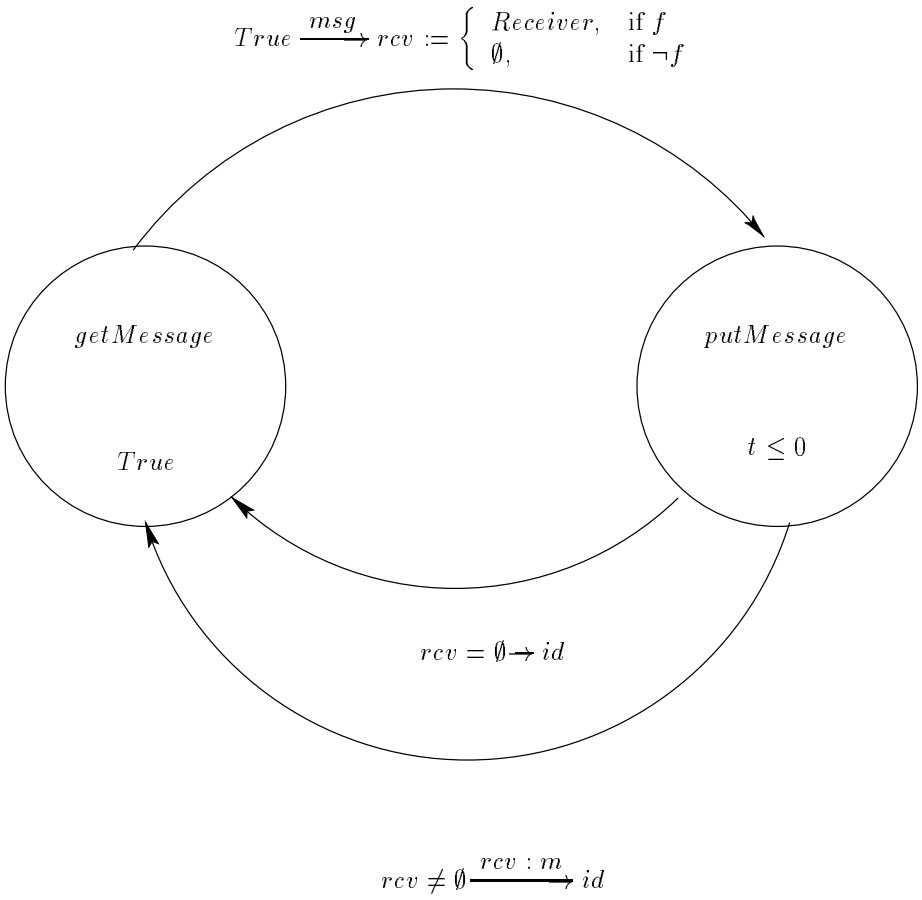

Figure 5: The communication channel: $Channel^{m}_{Xmitter \rightarrow Receiver}$

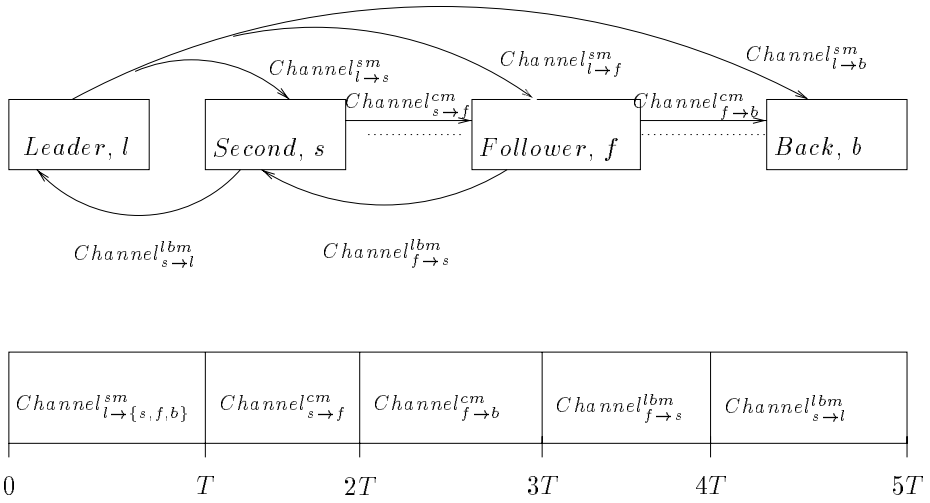

Figure 6: The platoon model.

Second Vehicle, s The second vehicle differs from other vehicles since it receives only one message from the leader, the sm.

Follower Vehicle, f The follower vehicle receives two messages: sm from the leader, and cm from second vehicle. It transmits cm to the back vehicle.

Back Vehicle, b The back vehicle is different from the rest of the vehicles since it does not transmit any control messages.

Each vehicle contains a controller and a diagnoser that runs as an application on a TDMA channel. Each cycle requires $5T$ time, where in a practical implementation $T = 20ms$ (we will use $T = 1$ for verification purposes). Messages of type sm and cm are transmitted by control applications and are received by both control and diagnoser applications. On the other hand, lbm messages are transmitted and received only by the diagnoser applications. As such, the role of lbm messages is to inflate the observation sets of the diagnosers.

At the beginning of each cycle the leader, l, broadcasts the synchronization message, sm. The order in which the message is received by the follower vehicles is not important. Vehicles other than the leader will not transmit or receive messages until they have successfully received sm. We will assume that the synchronization hardware of the radio devices of each vehicle is such that once they receive sm their clocks are synchronized for n cycles, during which they may communicate with each other.

After sm is transmitted by the leader, l, the second vehicle, s, transmits a control message to its follower, f. Then the follower, f, transmits its control message to the back vehicle, b, and this marks the end of the cycle for control message transmissions. Then 2 auxiliary messages are transmitted by the

diagnostic applications via the radio devices used by the controllers. First, f, transmits a loopback message to s then s transmits a loopback message to l.

4.2.1 Fault modeling

A fault can be modeled in several ways. Formally, a fault is a permanent breakdown of either the transmitter of a single radio device or its receiver. We will model transmitter faults by simply disengaging the outgoing channel[5] between that transmitter and its recipient. We will model a receiver fault by using a symbol to denote whether the receiver is operational or not. In the latter case, the receiver must not block the transmitter from transmitting a message into the channel, albeit the message will get lost. Figure 7 illustrates this design.

The transmitter, x, will attempt to send a message of type $m \in \{cm, sm, lbm\}$ to the receiver, r, at time cT where $c \in \{0, 1, 2, 3, 4, 5\}$. The 5 possible outgoing transitions are:

δ	Non faulty transmitter
δ^{f_x}	Faulty transmitter
γ	Success
γ^{f_r}	Faulty receiver, non faulty transmitter
γ^{f_x}	Faulty transmitter

Note that the successful transmission/reception of a message occurs only with transition γ, and also note that only successfully received messages are observable by a diagnoser. The observation is indicated by the propagation of the event $d_i : cm$ where d_i is the diagnoser of the receiving vehicle.

The auxiliary symbol $rcvFaultP$ (i.e. Receiver Faulty Predicate) is assigned a $\$True$[6] value whenever there is a permanent receiver fault. The reset function $\psi(t)$ resets the timer t:

$$\psi(t) := \begin{cases} 0, & \text{if } c = 5 \\ t, & otherwise \end{cases}$$

The auxiliary clock t' is used to reset the value of t to the correct value in case the timeout for t occurred at the end of the cycle.

We will continue with the formal description of each type of vehicle, l, s, f, b and we will use the transmitter and receiver of Figure 7 as a black box for a more compact representation.

4.2.2 Lead vehicle specification, l

The lead vehicle is the simplest of the four types of vehicles considered. In the first time slot (i.e. $c = 1$) it broadcasts the synchronization message to the

[5] communication channels are modeled as uni-directional.

[6] We will use \$ to indicate that a value is symbolic

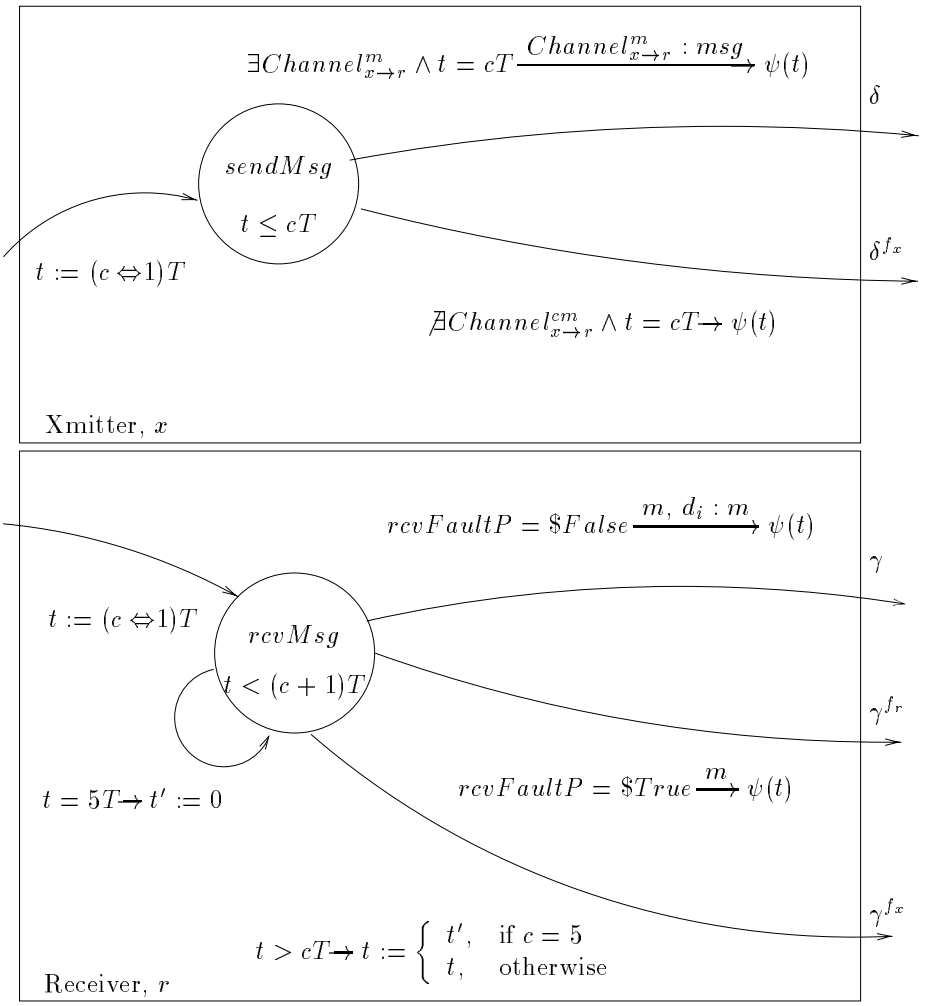

Figure 7: The transmitter/receiver fault model.

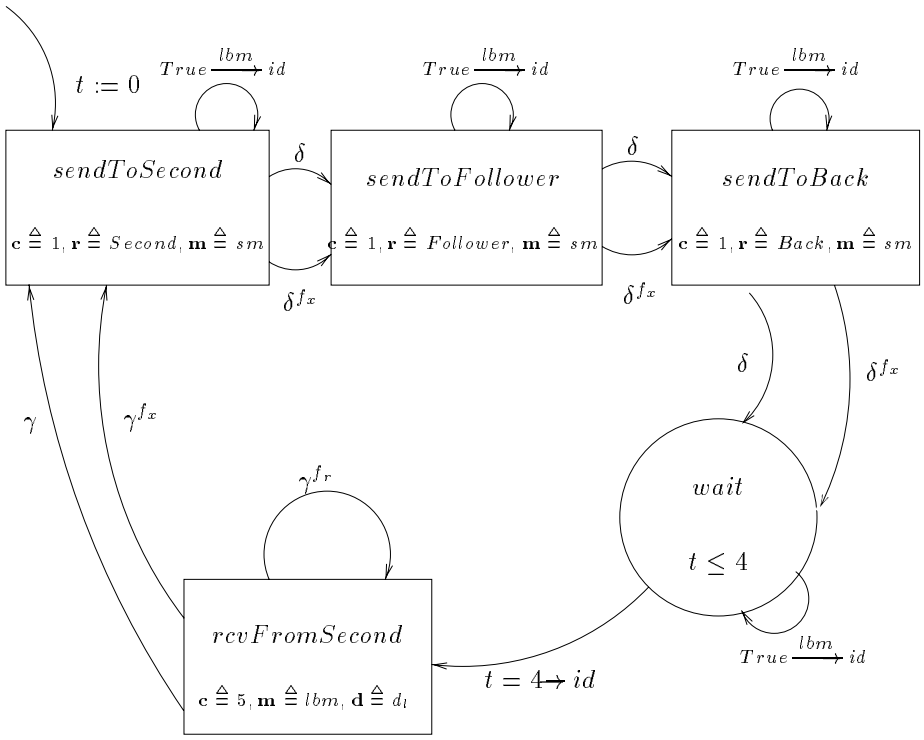

Figure 8: The lead vehicle plant model.

second, follower and back vehicle. Then it waits to receive a loopback message from the second vehicle in the last time slot (i.e. $c = 5$) and continues operation in the usual manner. Note that the broadcast is modeled as a series of uni-cast messages all occurring at the same time. Figure 8 formally specifies the leader vehicle plant model.

4.2.3 Second vehicle specification, s

The second vehicle initially waits for the synchronization message from the leader. Once sm is received from the leader, the synchronized radio device is assumed to maintain a synchronized clock for n cycles (i.e. n approximates the crystal drift). If $n > N$ (i.e. the clocks are not synchronized), the second vehicle must receive sm from the leader to continue normal operation, otherwise it remains $idle$ until the next cycle where the leader will broadcast sm once again. Note that this constraint on sm does not apply to cm or lbm type messages.

Figure 9 illustrates the second vehicle plant model. Note that each state maintains a self loop to avoid blocking the plant communication system.

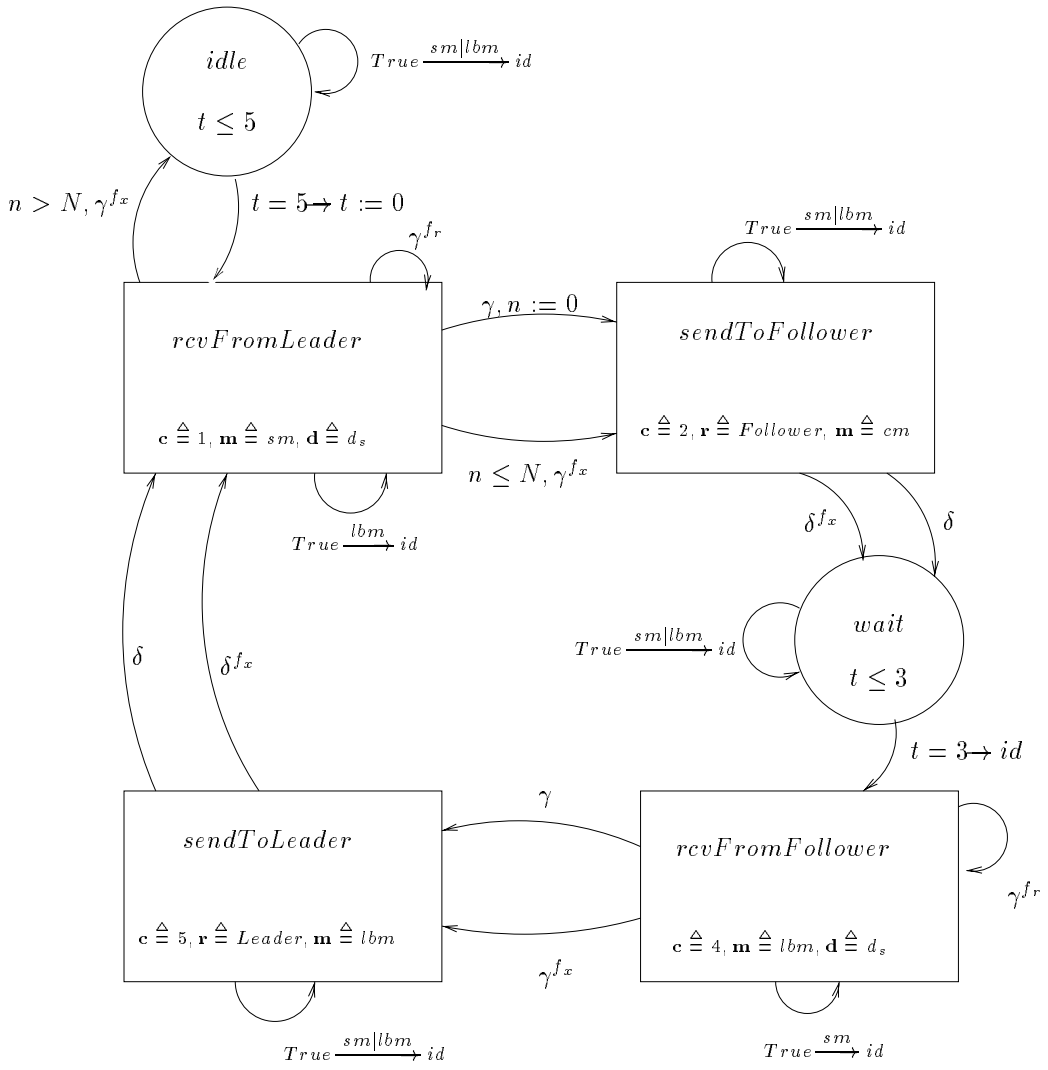

Figure 9: The second vehicle plant model.

4.2.4 Follower vehicle specification, f

The follower vehicle model is similar to that of the second vehicle model given in Section 4.2.3. It is illustrated in Figure 10.

4.2.5 Back vehicle specification, b

The back vehicle model is similar to that of the second vehicle model given in Section 4.2.3. It is illustrated in Figure 11.

4.3 Diagnoser design

We are interested in designing a diagnoser,d_i,$i \in \{l, s, f, b\}$, for each type of vehicle, l, s, f, b.

Notation 1 *Recall that a channel from vehicle x to vehicle r for transmitting m type messages is denoted with $Channel^{m}_{x \rightarrow r}$. Furthermore, the successful transmission of a message over this channel can only be observed by the receiver r upon the event γ (see Figure 7). We will use the notation $x : r$ to denote this certain event. Note that with this notation we have discarded the message type; i.e. the message type is immaterial for the diagnoser application.*

The plant failure events are $\Sigma_f = \{ltf, lrf, stf, srf, ftf, frf, brf\}$ where ltf stands for lead vehicle transmitter fault, lrf stands for lead vehicle receiver fault and so on. We will let any diagnoser identify any fault so that for all $x \in \{l, s, f, b\}$ $\Sigma_{mf_x} = \Sigma_m f = \Sigma_f$. The observation sets, Σ_{d_i}, for each diagnoser d_i are listed in Table 2. The event t is used to denote the passage of one time slot without any observations.

Diagnoser	*Observation Set*
d_l	$\Sigma_{d_l} = \{s : l, t\}$
d_s	$\Sigma_{d_s} = \{l : s, f : s, t\}$
d_f	$\Sigma_{d_f} = \{l : f, s : f, t\}$
d_b	$\Sigma_{d_b} = \{l : b, f : b, t\}$

Table 2: Observation sets for the diagnosers

We will choose the estimates of each diagnoser from 2^{Σ_f}; that is, for each diagnoser $\nu_i : \Sigma^*_{d_i} \rightarrow 2^{\Sigma_f}$. The decision function of the diagnosers will be chosen to be the same; i.e. for all $x \in \{l, s, f, b\}$ $\theta_x = \theta$ where $\theta : \Sigma_{m_l} \times \ldots \times \Sigma_{m_b} \rightarrow \Sigma_{mf}$.

4.3.1 The Global Picture

Figure 12 illustrates our discrete event abstraction for global plant behavior. The figure does not account for channel faults and is merely provided to illustrate the logic behind our design.

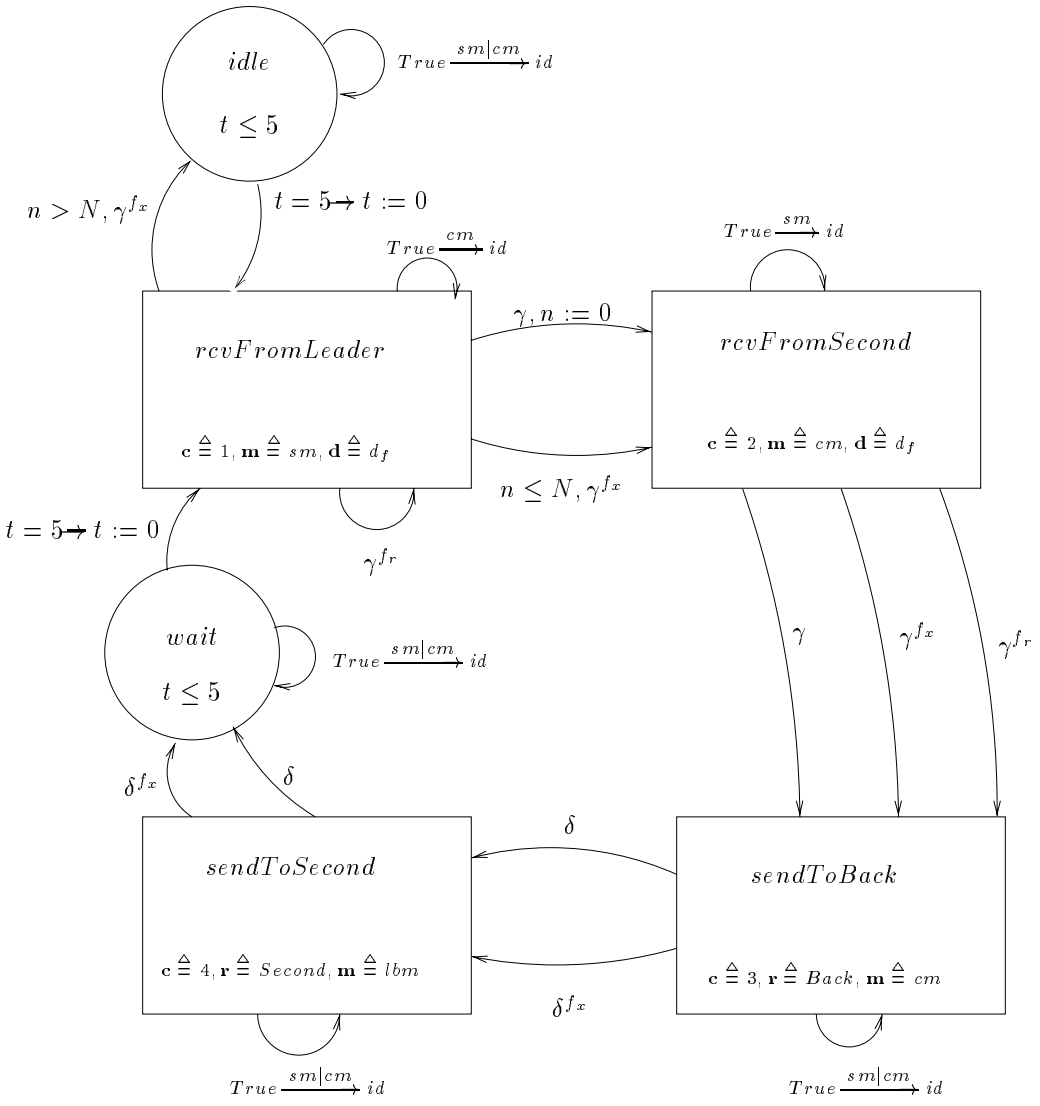

Figure 10: The follower vehicle plant model.

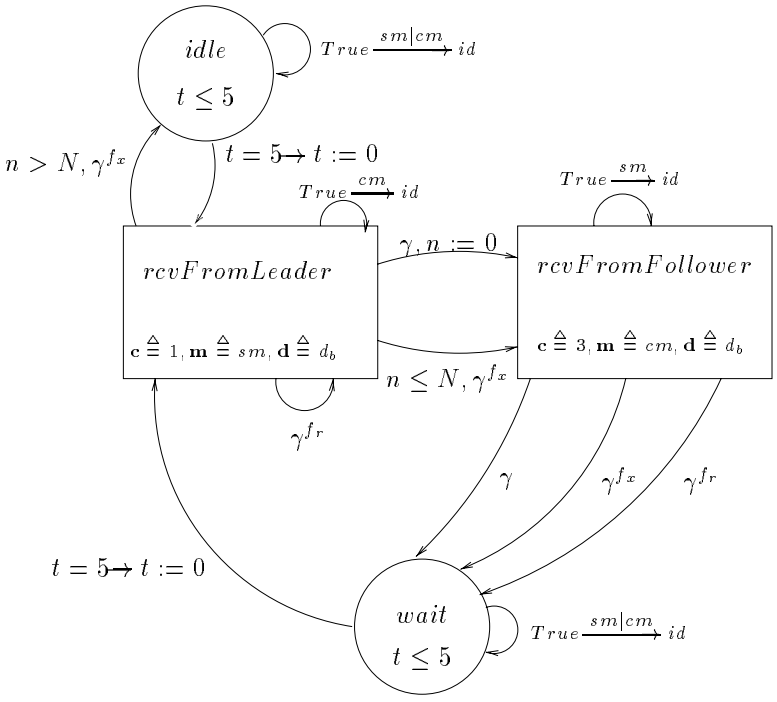

Figure 11: The back vehicle plant model.

Notation 2 *The modified regular expression* $(ab\ldots c)^*$ *will be understood as* $(ab\ldots c)^*(\lambda, a, ab, \ldots, ab\ldots c)$.

Figure 12 serves 3 purposes. First, it summarizes in a reasonable fashion the global behavior of the plant. Second, it illustrates the logic behind our design. That is, we base our design on the assumption that it is sufficient to consider the occurrence of each type of fault only at certain critical points in the abstract behavior of the plant. For example, we consider ltf and the resulting behavior of the plant only in state 1 where the leader is interested in transmitting. Similarly, we consider srf only when the second vehicle is interested in receiving a message. Third, the figure illustrates that a fault, $\sigma_f \in \{ltf, lrf, stf, srf, ftf, frf, brf\}$, occurring at time t can be mapped to the same fault occurring at an earlier time t_0 by means of appending a prefix in front of the resulting plant behavior that would have occurred at time t_0. This suggests that each diagnoser should generate estimates after making observations for one plant cycle. Since each plant cycle is 5 slots and only one observation is recorded per slot we will let the estimate functions to be an injective mapping from $\Sigma_{d_i}^{(}5)$ into Σ_{m_i}.

The following sections describe in detail the functions ν_i and the estimate sets Σ_{m_i} for each diagnoser and finally the decision functions θ.

4.3.2 Second vehicle diagnoser design

A simple design for the second vehicle diagnoser is to take the projection of the discrete event system of Figure 12 onto Σ_{d_s} and then to design Σ_{m_s}.

The simplified machine of this projection is given in Figure 13. One should readily be able to identify all possible trajectories of the abstract plant model of Figure 12 within this simplified machine. Furthermore, consider the regular operation of the plant under no faults. This operation is modeled as $(t\,l : s\,t\,t\,f : s)^*$. Under any kind of fault the resulting behavior will be $(t(l : s \mid t)tt(f : s \mid t))^*$. This is so because the occurrence of a fault can only disable the second vehicle from receiving a message. Thus the machine of Figure 13 captures all possible observations of the plant projected onto Σ_{d_s}.

The preceding discussion suggests, from symmetry, that we can parse observation sequences to simplify our observation structure. Table 3 lists the non-terminals (augmented events) a,b, c and d. With respect to these non-terminals we reduce the finite state machine in Figure 13 to a the much simpler machine of Figure 14. It suffices to observe the symmetry in the solid colored states to appreciate this reduction.

Now we are ready to design the estimates of the second vehicle diagnoser, i.e. Σ_{m_s}. We will argue in an ω-automata style fashion. That is, if the second vehicle diagnoser observes repeatedly many a's then it knows that there is nothing wrong with its receivers and the lead vehicle and follower vehicle transmitters. Note also that if the second vehicle can receive from the follower then the follower receiver must also be operating, otherwise it would not have been able to receive from the leader to synchronize its clocks. Hence, in the case of repeatedly many occurring a's the second vehicle diagnoser guesses that if there is a fault it must

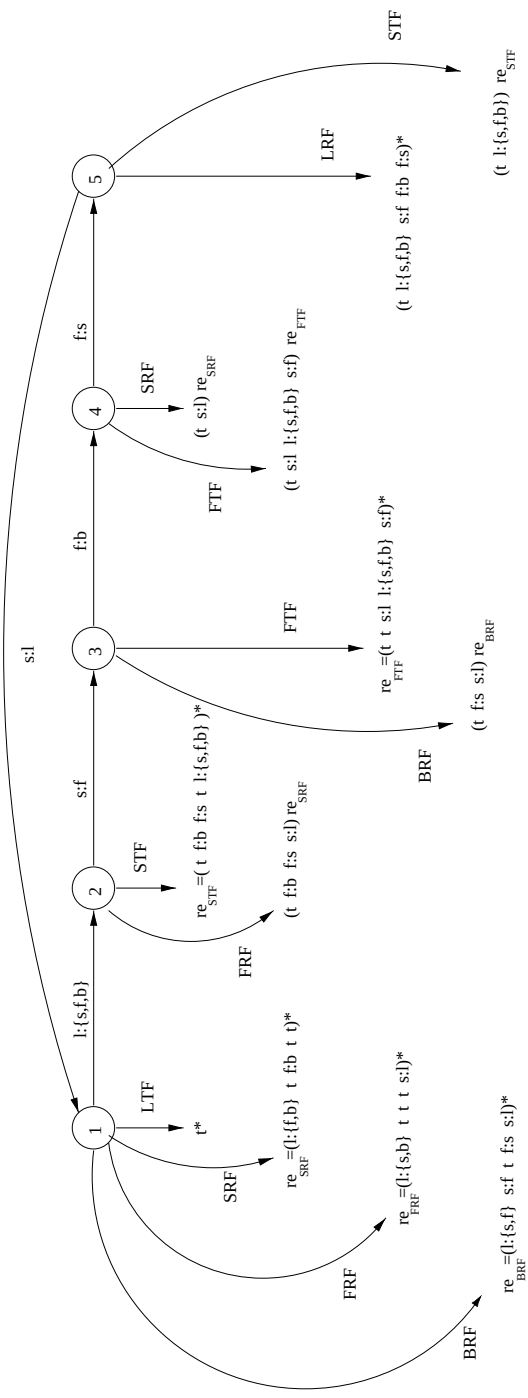

Figure 12: Discrete event abstraction for global plant behavior.

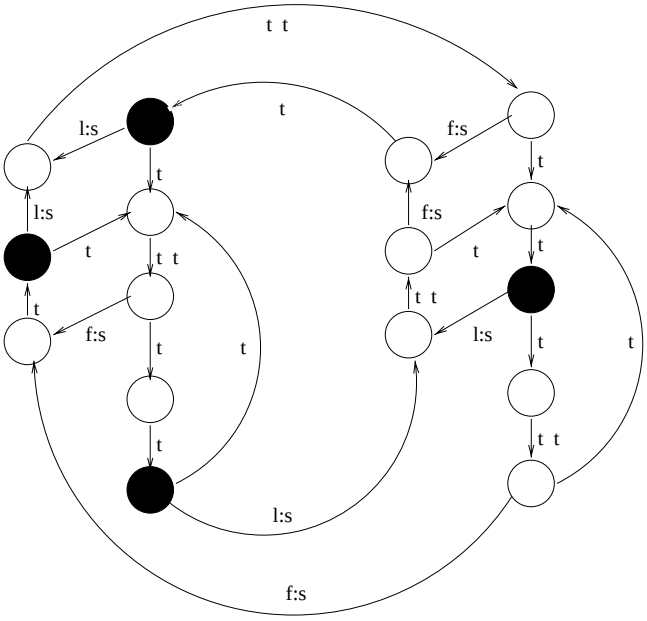

Figure 13: Second vehicle diagnoser observation structure.

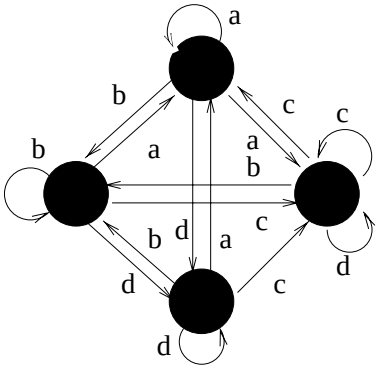

Figure 14: Reduced second vehicle diagnoser.

Non-terminal		*Observation sequence*
a	$=$	$l : s \;\; t \;\; t \;\; f : s \;\; t$
b	$=$	$t \;\; t \;\; t \;\; f : s \;\; t$
c	$=$	$l : s \;\; t \;\; t \;\; t \;\; t$
d	$=$	$t \;\; t \;\; t \;\; t \;\; t$

Table 3: Second vehicle parsed observations

be one of $\{lrf, stf, brf\}$. In the case of repetitive observation of b's the second vehicle assumes that

- either the lead vehicle transmitter is faulty and the follower vehicle is able to transmit on a previously synchronized clock (note that this means once a clock is synchronized it maintains its validity for sufficiently long)
- or there is a persistent channel fault in $Channel^{sm}_{l \to s}$.

In the first case the second vehicle diagnoser may safely assume that there is a fault and it is ltf. In the second case there may be no real hardware fault but the problem formulation of Section 2.1 states that a persistent channel fault may be identified as a hardware fault. Hence we will let the estimation set of repeatedly many occurring b's be $\{ltf\}$. In case of repeatedly many occurring c's it should be clear that the follower vehicle either has a transmitter or receiver fault. And in case of repeatedly many occurring d's it should be evident that either the lead vehicle is not transmitting or the second vehicle is not receiving.

We may modify these arguments to account for persistent or repetitive channel faults for all cases but this is not necessary since these types of faults will be identified wrongly as a hardware fault as we illustrated in the reasoning for the case of repeatedly many occurring b's. Table 4 summarizes the observation sets, parsed observations and estimates of the second vehicle diagnoser.

4.3.3 Lead, Follower and Back Vehicle Diagnosers

We will not explain in any detail the design of the lead, follower and back vehicle diagnosers. The design process is similar to that of the second vehicle. We would like to point out the only significant difference which occurs in the machinery for the lead vehicle. This machine differs from that of the others since the lead vehicle unlike the other vehicles receives only one message. The lead vehicle diagnoser observation structure is illustrated in Figure 15.

The follower and back vehicles share the same diagnoser observation structure with the second vehicle except that the a, b, c, d's are parsed differently. The following Tables 5,6,7 summarize the observation sets, parsed observations and estimates of the lead, follower and back vehicles.

This formulation requires existential quantification of n over the natural numbers. The number n is called the *decision time*. For verification purposes

i.

Diagnoser	*Observations Set*
d_s	$\Sigma_{d_s} = \{l:s, f:s, t\}$

ii.

Non-terminal		*Observation sequence*
a	$=$	$l:s \;\; t \;\; t \;\; f:s \;\; t$
b	$=$	$t \;\; t \;\; t \;\; f:s \;\; t$
c	$=$	$l:s \;\; t \;\; t \;\; t \;\; t$
d	$=$	$t \;\; t \;\; t \;\; t \;\; t$

iii.

Non-terminal, x	*Second vehicle estimate*, $\nu(x)$
a^n	$\{lrf, stf, brf\}$
b^n	$\{ltf\}$
c^n	$\{ftf, frf\}$
d^n	$\{ltf, srf\}$

Table 4: (i) Observation set for the second vehicle diagnoser, (ii) Second vehicle parsed observations, (iii) Second vehicle diagnoser estimates

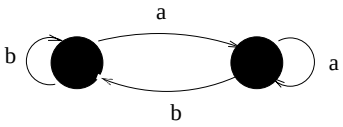

Figure 15: Reduced lead vehicle diagnoser.

we will initially make an attempt with $n = 1$ and if this does not work we will attempt with $n = 2$ and so on. Such a quantification implies that the verification of the diagnosers is a semi-decidable procedure.

4.3.4 Decision logic

Let the decision time be n. Then, each diagnoser will produce as estimate every n cycles. Let σ_{mf_l}, σ_{mf_s}, σ_{mf_f} and σ_{mf_b} be the estimates generated at cycle n. Then we choose θ to be the intersection of these estimates whenever the intersection yields a single fault:

$$\theta(\sigma_{m_l}, \sigma_{m_s}, \sigma_{m_f}, \sigma_{m_b}) = \begin{cases} g = \cap_{x \in \{l,s,f,b\}} \sigma_{m_x}, & \text{if } \|g\| = 1 \\ \emptyset, & otherwise \end{cases}$$

5 Modeling of diagnosers

Modeling the diagnosers requires modeling the parsers that identify the non-terminals a,b,c,d, and modeling the functions θ and ν_x for each $x \in \{l, s, f, b\}$ as

i.

Diagnoser	*Observation Set*
d_l	$\Sigma_{d_l} = \{s : l, t\}$

ii.

Non-terminal		*Observation sequence*
a	$=$	$l : s \;\; t \;\; t \;\; f : s \;\; t$
d	$=$	$t \;\; t \;\; t \;\; t \;\; t$

iii.

Non-terminal, x	*Lead vehicle estimate*, $\nu(x)$
a^n	$\{ftf, frf, brf\}$
b^n	$\{ltf, lrf, stf, srf\}$

Table 5: (i) Observation set for the lead vehicle diagnoser, (ii) Lead vehicle parsed observations, (iii) Lead vehicle diagnoser estimates

timed automatons. We will not present these models here since their purpose is understood from Tables 5,4,6 and 7.

6 Verification of diagnosability

We must first show that our design is admissible. The design is clearly *consistent* since we are taking the synchronous composition of the plant model and the diagnoser models. The *passivity* of our design is verified automatically. We embellish each diagnoser design such that there is an auxiliary transition from each state in each diagnoser to an *error* state. Say that S is a discrete state of diagnoser d_i and that it has the outgoing transitions:

$$\begin{aligned} t_1 &= g_1 \xrightarrow{e_1} a_1 \\ t_2 &= g_2 \xrightarrow{e_2} a_2 \\ &\vdots \\ t_n &= g_n \xrightarrow{e_n} a_n \end{aligned}$$

we will introduce a new outgoing transition, t_{n+1}, to the state S:

$$t_{n+1} = \neg g_1 \wedge \neg g_2 \wedge \ldots \wedge \neg g_n \rightarrow id$$

that takes the machine into the *error* state. Once an auxiliary transition is added to each of the discrete states in each of the diagnosers we check for the safety property:

$$True \;\; \Rightarrow \;\; \neg\exists\Diamond error \qquad (1)$$

i.

Diagnoser	*Observation Set*
d_f	$\Sigma_{d_f} = \{l : f, s : f, t\}$

ii.

Non-terminal		*Observation sequence*
a	$=$	$l : s \;\; t \;\; t \;\; f : s \;\; t$
b	$=$	$t \;\; t \;\; t \;\; f : s \;\; t$
c	$=$	$l : s \;\; t \;\; t \;\; t \;\; t$
d	$=$	$t \;\; t \;\; t \;\; t \;\; t$

iii.

Non-terminal, x	*Follower vehicle estimate*, $\nu(x)$
a^n	$\{lrf, ftf, brf\}$
b^n	$\{ltf\}$
c^n	$\{stf, srf\}$
d^n	$\{ltf, frf\}$

Table 6: (i) Observation set for the follower vehicle diagnoser, (ii) Follower vehicle parsed observations, (iii) Follower vehicle diagnoser estimates

That is, if the *error* state is not reachable then the diagnosers do not block the plant. Furthermore, it is immediate from the observation structures for each diagnosers that they do not force any plant events and hence the design is *passive*.

Causality of the design should also be clear from Definition 2. Our choice of the estimate functions, ν_i for each diagnoser are injective, hence our design is causal.

6.1 Correctness of the diagnoser design

To verify the correctness of the design we must verify:

1. that there are no *missed detections*. If a fault, $\sigma_f \in \{ltf, \ldots, brf\}$, occurs then any possible behavior in the design L must eventually lead to state in which $\theta(\sigma_{m_l}, \ldots, \sigma_{m_b}) = \sigma_f$, where σ_{m_x}, $x \in \{l, s, f, b\}$, are the diagnoser estimates.

2. that there are no *false alarms*. If there is no fault then it is no plant behavior that leads to state in which $\theta(\sigma_{m_l}, \ldots, \sigma_{m_b}) \neq \emptyset$.

We will enrich the plant model with a *fault-generation* unit such that a selected fault, σ_f, may non-deterministically be generated at any time during a cycle. Figure 16 illustrates this design. We take note of our fault model in Section 4.2, i.e. transmission faults are modeled by disengaging the outgoing

i.

Diagnoser	*Observation Set*
d_b	$\Sigma_{d_b} = \{l:b, f:b, t\}$

ii.

Non-terminal		*Observation sequence*
a	$=$	$l:s \;\; t \;\; t \;\; f:s \;\; t$
b	$=$	$t \;\; t \;\; t \;\; f:s \;\; t$
c	$=$	$l:s \;\; t \;\; t \;\; t \;\; t$
d	$=$	$t \;\; t \;\; t \;\; t \;\; t$

iii.

Non-terminal, x	*Back vehicle estimate*, $\nu(x)$
a^n	$\{lrf, stf, srf\}$
b^n	$\{ltf\}$
c^n	$\{ftf, frf\}$
d^n	$\{ltf, brf\}$

Table 7: (i) Observation set for the back vehicle diagnoser, (ii) Back vehicle parsed observations, (iii) Back vehicle diagnoser estimates

uni-directional channel of that transmitter and receiver faults are modeled symbolically. There is no loss of generality in forcing the fault to occur during the first cycle since the plant operation is periodic.

The function θ is modeled by the timed automaton of Figure 17. And finally we can specify conditions 1 and 2 formally with TCTL:

$$wait \Rightarrow \forall\Diamond \;\; Good\,Decision \tag{2}$$

$$wait \Rightarrow \forall\Box \;\; \neg Bad\,Decision \tag{3}$$

$$\not\exists\sigma_f \Rightarrow \forall\Box \;\; Detect \tag{4}$$

6.2 Verification of diagnoser correctness with the SHIFT-Kronos connection

We must be able to convert Equations 2,3 and 4 into reachability specifications. It is clear that Equations 3 and 4 are reachability problems and they can be re-written in a format accepted by our verifier, KRONOS:

$$wait \Rightarrow \neg\exists\Diamond \;\; Bad\,Decision \tag{5}$$

$$\not\exists\sigma_f \Rightarrow \neg\exists\Diamond \;\; \neg Detect \tag{6}$$

However, it is not possible to re-write Equation 2 in any way as to make it a reachability problem. Thus we will attempt to verify Equations 3 and 4 with the above formulation and once this is accomplished we will replace the timed

$$t < 5T \rightarrow \begin{cases} Channel^{sm}_{l \rightarrow \{s,f,b\}} = \emptyset, & \text{if } \sigma_f = LTF \\ Channel^{cm}_{s \rightarrow f}, Channel^{lbm}_{s \rightarrow l} = \emptyset, & \text{if } \sigma_f = STF \\ Channel^{cm}_{f \rightarrow b}, Channel^{lbm}_{f \rightarrow s} = \emptyset, & \text{if } \sigma_f = FTF \\ rcvFaultP(l) = \$True, & \text{if } \sigma_f = LRF \\ rcvFaultP(s) = \$True, & \text{if } \sigma_f = SRF \\ rcvFaultP(f) = \$True, & \text{if } \sigma_f = FRF \\ rcvFaultP(b) = \$True, & \text{if } \sigma_f = BRF \end{cases}$$

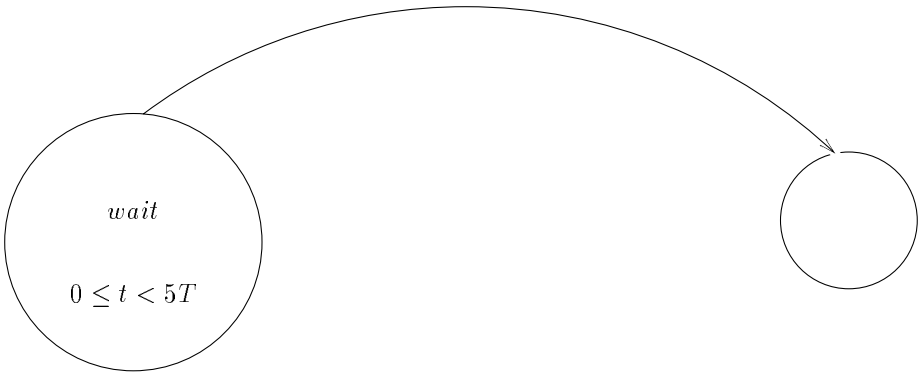

Figure 16: Fault generation unit for a given fault σ_f.

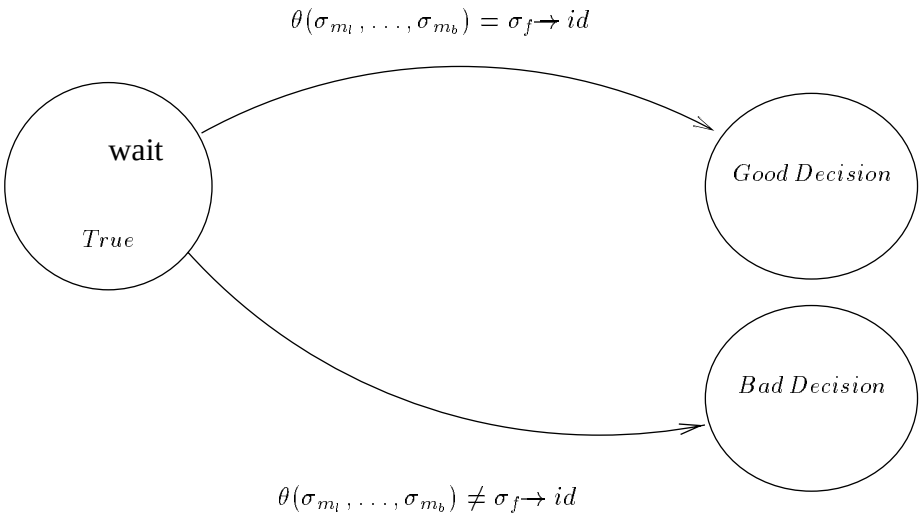

Figure 17: Timed automaton for modeling the function $\theta(\sigma_{m_l}, \sigma_{m_s}, \sigma_{m_f}, \sigma_{m_b})$ given a fault σ_f.

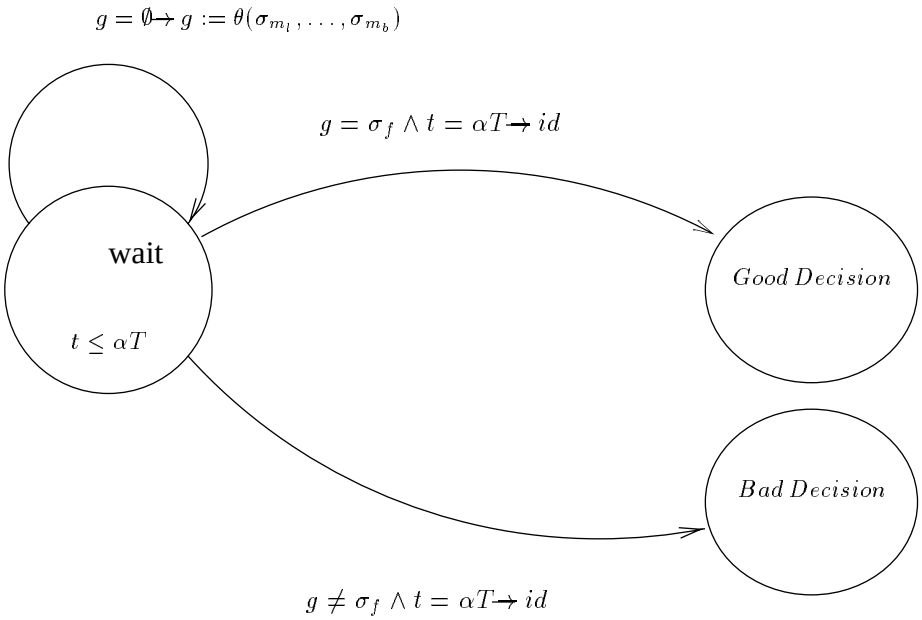

Figure 18: Modified Timed automaton for modeling the function $\theta(\sigma_{m_l}, \sigma_{m_s}, \sigma_{m_f}, \sigma_{m_b})$ given a fault σ_f.

automaton of Figure 17 with that of the timed automaton of Figure 18 to verify an equivalent expression to that given in Equation 2.

Let us investigate the new machinery we have developed for the verification of Equation 2. The machine continually computes θ until it makes a decision, which it saves in the auxiliary variable g. At some time $t = \alpha T$ we force the machine to execute either one of two transitions whose guards are such that, $g = \sigma_f \cup g \neq \sigma_f = I$ (i.e. either one of both transitions are always allowed at time $t = \alpha T$). With this machine we would like to verify the following property:

$$\exists \alpha \in N \ [wait \ \ \Rightarrow \ \ \neg \exists \Diamond Bad \ Decision \ ] \tag{7}$$

KRONOS will not quantify α hence we must manually perform the quantification. If we can find an α, say α_0, such that the reachability property of Equation 7 is verified, then, it is clear that any choice of $\alpha \geq \alpha_0$ will imply that the same reachability property is verified. Formally, we call the minimal such α the *settling time*, denoted with α^*:

$$\alpha^* = \arg\min_{\alpha \in N} \ [wait \ \ \Rightarrow \ \ \neg \exists \Diamond Bad \ Decision \ ] \tag{8}$$

7 Conclusions

We performed two experiments for each fault $\sigma_f \in \{ltf, lrf, stf, srf, ftf, frf, brf\}$.

The first experiment is to test for the semi-decidable reachability property of Equation 7. This is a semi-decidable procedure because we need to quantify the *decision time*, n (see Section 4.3.3), and the *settling time*, α (see Section 6.2), over the natural numbers. For purposes of verification we have disabled channel faults in the verification. This suggested that the decision time, n, should be 1. As for the settling time, α, we had a trial and error process in which we found that with $\alpha = 15$ (i.e. 15 time slots equals 3 cycles) the verification terminated with $Bad\,Decision$ not reachable (i.e. $Good\,Decision$ is reached due to the modified machine of Figure 18). With $n = 1$ and $\alpha = 15$ the verification required approximately 2650 symbolic states (corresponding to 350 $\Leftrightarrow$400 MB of RAM) with a termination time of 3+ minutes on an Ultra SPARC-2 Enterprise system running Solaris.

The second experiment is for the verification of the reachability properties of Equations 5 and 6. These are decidable once the decision time, n, is chosen. Since we used $n = 1$ in the first experiment we use $n = 1$ in the verification of Equations 5 and 6. Since the reachability property of Equaiton 5 is implied by the modified reachability property of Equation 7 (i.e. that of the first experiment), we only executed kronos on the reachability property of Equation 6. This safety property was verified.

We will not discuss the use of KRONOS and the specific parameters needed to duplicate these experiments in these report. The source code for our design is readily available from [20].

References

[1] Raja Sengupta. Diagnosis and communication in distributed systems. Technical report, California PATH, University of California, Berkeley, CA, July 1998.

[2] F. Eskafi. A diagnostic system design for the intra-platoon communication system in NAHSC demo'97. Technical report, California PATH, University of California, Berkeley, CA, December 1997. preprint.

[3] A. E. Lindsey and P. Vishwanath. Design, verification and failure diagnosis of wireless communication protocols for AHS. *In* Proc. 1997 IEEE Conference of Intelligent Transportation Systems, November 1997.

[4] R. Rajamani, J.K. Hedrick, and A. Howell. A complete fault diagnostic system for longitudinal control of automated vehicles. *In* Proc. Symposium of Advanced Automotive Technologies, 1997 ASME International Congress, 1997.

[5] H. T. Şimşek. Shift Tutorial: A first course for SHIFT programmers. Technical report, University of California, Berkeley, CA, February 1999. preprint.

[6] A. Deshpande, A. Göllü, and L. Semenzato. The SHIFT programming languange and run-time system for dynamic networks of hybrid automata. *IEEE Trans. Automatic Control special issue on Hybrid Systems*, April 1998.

[7] A. Deshpande, A. Göllü, and L. Semenzato. SHIFT: Reference manual. Technical Report 10856, California PATH, University of California, Berkeley, CA, 1997. PATH research report, UCB-ITS-PRR-97-8.

[8] SHIFT Team. `http://www.path.berkeley.edu/shift`. world wide web. SHIFT home page.

[9] S. Yovine. Verification and implementation of SHIFT models. Technical report, California PATH, University of California, Berkeley, CA, 1998.

[10] S. Yovine. Kronos: A verification tool for real-time systems. *In Springer International Journal of Software Tools for Technology Transfer*, 1(1+2), October 1997.

[11] C. Chen, M. Asaw, and B. Foreman. Outdoor measurements on WaveLAN radio. Technical Report 9384, 96-2, California PATH, University of California, Berkeley, CA, December 1995.

[12] C. Chen and B. Foreman. A discussion of the WaveLAN radio as relevant to automated vehicle control systems. Technical Report 9383, 96-1, California PATH, University of California, Berkeley, CA, 1996.

[13] B. Foreman. A survey of wireless communication technologies for automated vehicle control. *In* Proc. SAE Future Transportation Technology Conference*, Costa Mesa, CA*, August 1995.

[14] PATH wireless vehicle communication system: Overview and functional specifications. Technical report, MPI Corp, Atlanta, GA, October 1996.

[15] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proc. IEEE*, pages 77(1):81–98, January 1989.

[16] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Trans. Automatic Control*, page 40, September 1995.

[17] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Failure diagnosis using discrete-event models. *IEEE Trans. Control Systems Technology*, page 4, March 1996.

[18] R. Boubour, C. Jard, A. Aghasaryan, E. Fabre, and A. Benveniste. A Petri net approach to fault detection and diagnosis in distributed systems (part 1). *In* Proc. 36th IEEE Conf. on Decision and Control*, San Diego, CA*, December 1997.

[19] A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour, and C. Jard. A Petri net approach to fault detection and diagnosis in distributed systems (part 2). *In* Proc. 36th IEEE Conf. on Decision and Control*, San Diego, CA*, December 1997.

[20] H. T. Şimşek. `http://www.eecs.berkeley.edu/ simsek`. world wide web. SHIFT/Kronos code for intra-platoon communication diagnoser design.