Title
A Specification Of An Automated Freeway With Vehicle-borne Intelligence

Permalink
https://escholarship.org/uc/item/67b3b8zv

Author
Hitchcock, Anthony

Publication Date
1992
This paper has been mechanically scanned. Some errors may have been inadvertently introduced.
A Specification of an Automated Freeway with Vehicle-Borne Intelligence

Anthony Hitchcock

UCB-ITS-PRR-92-18

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 and National Highway Traffic Safety Administration.

The contents of this report reflect the views of the author who is responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the State of California. This report does not constitute a standard, specification, or regulation.
CONTENTS

List of Figures iii
Glossary of Terms iv

1. Introduction 1
2. Some Definitions 2
3. Hazards 3
4. System Architecture 4
   4.1 Architecture, Level 5: Law 4
   4.2 Architecture, Level 4: Network 6
   4.3 Architecture, Level 3: Link 6
   4.4 Architecture, Level 2: Platoon 6
   4.5 Architecture, Level 1: Regulation 8
   4.6 Architecture, Level 0: Physical 10
5. Physical Layout 10
6. System Modes 11
7. Faulty Vehicles 13
8. Operation of the System 14
   8.1 Manoeuvres 15
      8.1.1 Merge 15
      8.1.2 Split 17
      8.1.3 Change-Lane 19
      8.1.4 Emergency-Change 21
      8.1.5 Forced-Split 23
   8.2 Probes 24
      8.2.1 Platoon-Leader Probe 24
      8.2.2 In-Platoon Probe 26
   8.3 Entry and Exit 28
   8.4 Operation of the Vehicle-Borne Controller 28
   8.5 Operation of the Roadside Controller 30
Acknowledgements 31
References 32
Appendixes

A.1 Introduction A-1
A.2 How to Use the Module Definitions A-1
A.3 General A-3
A.4 Supervisor A-7
A.5 Busycheck A-13
   A.5.1 Merge A-13
   A.5.2 Split A-19
   A.5.3 Change-Lane A-25
   A.5.4 Forced-Split A-42
   A.5.5 Emergency-Change A-57
A.6 Platoon-Leader’s Probe A-68
A.7 Receipt of Messages A-73
A.8 New Faults A-79
A.9 New Actions A-83
A.10 Control Data A-87
A.11 Joining and Leaving A-90
A.12 System A-94
A.13 List of Messages A-98
List of Figures

Figure 1. IVHS Control Architecture (after Varaiya and Shladover). 5
Figure 2. Merge (after Hsu, et al., 1991). 16
Figure 3. Split (after Hsu, et al., 1991). 18
Figure 4. Change-Lane. 20
Figure 5. Emergency Change-Lane. 22
Figure 6. Platoon Leader Probe. 25
Figure 7. In-Platoon Probe. 27
Figure A. 1. Supervisor. A-6
Figure A.2. Merge. A-14
Figure A.3. Split and Split-change-lane. A-20
Figure A.4. Change Lane (initiator). A-26
Figure A.4a. Change Lane (recipient). A-27
Figure A.5. Forced Split (initiation). A-43
Figure A.5a. Forced Split (implementation). A-44
Figure A.6. Emergency Change (initiator). A-59
Figure A.6a. Emergency Change (participant). A-60
Figure A.7. Forward Probe. A-69
Figure A.8. Messages. A-74
Figure A.8a. Messages (continued). A-75
Figure A.9. Fault Flags. A-80
Figure A.10. Action Calls. A-84
Figure A.11. Control Data. A-88
Figure A.12. Joining and Leaving. A-91
GLOSSARY OF TERMS

In this paper, a number of specially-defined terms of art are used, which include abbreviations. The following table refers to the definitions of the terms and spells out the abbreviations.

<table>
<thead>
<tr>
<th>Term</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL = automated lane</td>
<td>1</td>
</tr>
<tr>
<td>AL license</td>
<td>4.1</td>
</tr>
<tr>
<td>AR = asynchronous record</td>
<td>4.4</td>
</tr>
<tr>
<td>b.e.m. = block entry marker</td>
<td>4.5</td>
</tr>
<tr>
<td>block</td>
<td>4.5</td>
</tr>
<tr>
<td>C mode = Crashstop</td>
<td>6</td>
</tr>
<tr>
<td>CA mode = Closed-Ahead</td>
<td>6</td>
</tr>
<tr>
<td>dormitory</td>
<td>4.1</td>
</tr>
<tr>
<td>fence</td>
<td>5</td>
</tr>
<tr>
<td>follower</td>
<td>2</td>
</tr>
<tr>
<td>free agent</td>
<td>2</td>
</tr>
<tr>
<td>full platoon braking</td>
<td>2</td>
</tr>
<tr>
<td>gate</td>
<td>5</td>
</tr>
<tr>
<td>loc = location = odometer reading and lane #</td>
<td>4.5</td>
</tr>
<tr>
<td>manual spacing</td>
<td>2</td>
</tr>
<tr>
<td>MON = self-monitor</td>
<td>4.5</td>
</tr>
<tr>
<td>N mode = Normal</td>
<td>6</td>
</tr>
<tr>
<td>NE mode = No-Entry</td>
<td>6</td>
</tr>
<tr>
<td>platoon leader</td>
<td>2</td>
</tr>
<tr>
<td>platoon spacing</td>
<td>2</td>
</tr>
<tr>
<td>R mode = Resume</td>
<td>6</td>
</tr>
<tr>
<td>S mode = Stop</td>
<td>6</td>
</tr>
<tr>
<td>SA mode = Slow-Ahead</td>
<td>6</td>
</tr>
<tr>
<td>TL = Transition lane</td>
<td>4.6</td>
</tr>
<tr>
<td>turning-point</td>
<td>4.6</td>
</tr>
<tr>
<td>VPD = Vehicle Presence Detector</td>
<td>4.6</td>
</tr>
<tr>
<td>vsv = Vehicle-borne State Vector</td>
<td>4.4</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

This paper is complete in itself. However, along with another paper (Hitchcock, 1992b) it can also be regarded as an appendix to “Methods for Analysis of IVHS Safety: Final Report of PATH MOU 19” (Hitchcock, 1992a). Readers not familiar with the area are strongly advised to read the other report first. An even shorter account of the background can be found in Hitchcock 1991.

Further, the work here extends the work of Hsu, et al. (1991). The work plan of the whole study included two worked examples of automated freeways. The overall objective was to demonstrate and refine the techniques of specification and safety analysis. The two examples should therefore be very different. The first was a single-AL system (AL = automated lane). Intelligence was concentrated in the infrastructure. It was therefore natural to look at a multi-AL system with mainly vehicle-borne intelligence.

Hsu and her colleagues had quite different concerns. Their purpose was to investigate a method of proving conformity of a conceptual design to its specification. This method used the logical technique enshrined in a computer program called COSPAN. The example they used to make their tests concrete was a partial design of an automated freeway. The freeway had many lanes. Intelligence was entirely vehicle-borne in the lower levels of the architectural hierarchy (see Section 4). This work was also part of the PATH program. It seemed sensible to build on Hsu’s design in the present work. This was particularly the case since Hsu and her colleagues were able to show that their design did meet their partial specification. Accordingly, the permission of the authors was sought. The design proposed by Hsu, et al. (1991) is close to being a subsystem of the design set out here. However, Hsu’s design is incomplete for the present purpose. We have extended it very considerably. Some changes have also been made to what Hsu, et al. (1991) proposed.

The specification for an automated freeway is set out in a fully formal manner in this paper. Subsequently, a series of safety analyses was carried out on the specification. These are reported elsewhere (Hitchcock, 1992b). The objective of the programme of work, of which this is part, is to derive a technique of safety analysis for such systems. The system reported here is the second example on which a trial analysis has been demonstrated. The analysis depends on the precise nature of the system specified. It is therefore necessary that the specification be recorded without possibility of ambiguity. Great detail is thus required. Also, the method of analysis recommended by Hitchcock (1992a) does require formal documentation. This paper is consequently intended as an exemplar of such documentation. This applies especially to the
appendixes. In the appendixes a formalized language has been used, which is analogous to some computer languages.

This report specifies the second example. To demonstrate the method of safety analysis a procedure called fault tree analysis was then applied. The fault tree for the present example is discussed in Hitchcock, 1992b.

The freeway specified here operates with vehicles in platoons. This is the basis on which other work in PATH has been carried out. There was no good reason to do other here. The safety argument in favour of platoons is reviewed in Hitchcock, 1992a.

The method of demonstrating safety used in the larger programme starts by defining certain hazards. A safety criterion is selected. The criterion used here is that two or more simultaneous faults must occur independently before the hazards can arise. The process is described in Hitchcock, 1991. Hazards are detailed later (see Section 3).

In what follows, we describe the whole design. Later, a short section (see Section 9) distinguishes the part that is due to Hsu and her colleagues from what is new.

2. SOME DEFINITIONS

Each vehicle will have a maximum deceleration when brakes are applied, which will vary between vehicles. Under some circumstances it will be necessary to decelerate as quickly as possible, short of generating within-platoon collisions. The appropriate deceleration is called full platoon braking. It is clearly a function of the road surface condition, and is therefore set by each block as vehicles enter.

Platoon spacing is that spacing within which a vehicle decelerating at full platoon braking can avoid colliding with one ahead when the latter is decelerating at some standard greater rate (one might choose 0.8 g — a practical maximum). The following platoon is supposed to be warned within some standard interval.

Manual spacing is the spacing at which drivers normally drive without being alarmed. It is a function of vehicle speed and road surface condition. A parameter to describe the latter is passed by the block controllers on entry.

On ALs, vehicles move in platoons. The first is the platoon leader; others are followers. A one-vehicle platoon is called a free agent.
3. HAZARDS

A hazard is defined as a situation in which if one further fault occurs, a high speed collision may ensue. The safety criterion as explained in Hitchcock, 1992a is that a hazard will not arise without two simultaneous faults of system components. What we have to guard against are crashes arising from the way the system is designed. Not all accidents are of this kind. If, in particular, a driver causes his/her vehicle to collide with an automated one in a place where the presence of both is permitted, this is not a hazard.

High-speed collisions may occur:

a. when all platoons involved are under control, automated or manual (a free agent is a one-member platoon). In this case vehicles were too close to each other before a final control failure.

b. when one platoon is not under control. This will happen if automatic control is switched off before the driver is ready, or not switched on when the driver lets go.

In case a above, we are only concerned when the rear platoon is under automatic control. In this case (remembering that a free agent is a one-member platoon), a further failure will cause a catastrophe if:

**Hazard 1.** A platoon is separated from one ahead of it, or from a stationary object in its path, by less than platoon spacing; or

**Hazard 2.** A vehicle, not under system control, is at an unmeasured or unknown distance in front of a platoon.

Case b above refers to the case where neither driver nor system is in fact controlling the vehicle, or where a driver is placed in a situation where he/she cannot control it.

**Hazard 3.** A vehicle is released to manual control before the driver has given a positive indication that he/she accepts it.

**Hazard 4.** A vehicle is released to manual control at less than manual spacing from the vehicle ahead of it; or at such a relative speed that manual spacing will be realized in less than 2 seconds; or while the brakes are being applied.

Danger may arise if a driver tries to surrender control before the system accepts it. This is an aberration of a manual driver, but, by definition, it does not give rise to hazards.
4. SYSTEM ARCHITECTURE

An AVCS system architecture of a system similar to this one has been described by Varaiya and Shladover (1991). This architecture is more general than the authors claim, and we shall use Varaiya and Shladover’s language to describe the architecture of the present system. The architecture is illustrated in Figure 1. It is hierarchical. Also, it is composed of modules. Modules may be hardware or software. The modules communicate with others at the same level in the hierarchy, and also with modules one level up and one level down. Their interfaces are fully specified. The significance of this is that if an improved module design is subsequently produced it can be slotted into the system without redesigning of the whole. In particular, it becomes possible to replace a single non-safety critical module with impunity. A total reappraisal of safety is not required.

The lower levels of the hierarchy are responsible for control of vehicle movement. They are safety-critical, as Figure 1 indicates. The highest level (this is an addition to the Varaiya and Shladover description) is law.

4.1 Architecture, Level 5: Law

The relevant law will control operations of automated freeways in many cities. Many of the problems which will arise are not covered by the present law. Here we shall state only the law, common to all city systems, which is required for operation. There are also a variety of wider areas where legal or quasi-legal regulation will be required. These include the mechanism for setting standards and ensuring or certifying conformity with them. We assume that the law will say:

a. Every vehicle trying to enter the AL will bear a remotely readable record which certifies that it is equipped with equipment of appropriate design. There will be a validation bit which indicates that the equipment has not been diagnosed as faulty, either by the vehicle itself or by the system. This bit, together with some data about the vehicle, will be called the AL, license. Only vehicles with a valid AL license may enter or remain on the ALs. Only a suitably qualified person may revalidate an invalid AL license. If a license becomes invalid while on the ALs (i.e., if the vehicle develops a fault) the vehicle should quit the ALs at once.

b. It will be an offence to fail to resume manual control on exit from the ALs. If a request to do so is disobeyed, the vehicle may be re-admitted to the system and carried to some point where it is safe to leave it. (Periodically there will be areas where such vehicles can be parked. We call them dormitories.)
Figure 1. IVHS Control Architecture (after Varaiya and Shladover).
c. It will be an **offence** to enter the system carrying an external or ill-secured load, or with a trailer not itself equipped with communication devices.

d. “Hacking,” that is, emitting signals designed to influence the system to do something which it would not normally do, is illegal. This applies particularly to manually-controlled vehicles which declare themselves to have valid licenses.

4.2 Architecture, Level 4: Network

This controls more complex aspects of routing in the light of overall flows on the network. It also sets the parameters which correct the values of platoon spacing, etc., in the light of prevailing weather conditions. Physically, the network controller is infrastructure based. It will be connected to the link-level controllers, probably by hard-wiring.

4.3 Architecture, Level 3: Link

Link level prescribes the route to be followed by a vehicle through the system. The desired route is stored in the vehicle. The choice is informed both by the data from the network level and the destination stated by the vehicle on entry. When the time comes to change lanes, the lower levels read the internal record, and determine if local conditions permit the manoeuvre. If they do not, the change is made later. Under some conditions (e.g., lane closures) route may be updated after entry. The efficiency of the link level algorithms clearly affects the capacity of the ALs. Efficiency here is thus of economic importance.

Link level controllers are infrastructure-based. There will be one for every 5-20 miles of freeway. They can communicate directly (presumably at high or optical frequencies) with platoon-level controllers on vehicles in their area, and also with the platoon-level infrastructure-based controllers. This latter link is probably accomplished by hard-wiring. The platoon-level controllers at the roadside are effective only in fault conditions. Data are exchanged about lane closures, emergency operations, and the like. It is at this level, for example, that human supervisors or the Highway Patrol will intervene if it is necessary to manoeuvre an emergency vehicle on to the ALs.

Link level does need to know which vehicles are in its area and where they are going. However, this is not a safety-critical function.

4.4 Architecture, Level 2: Platoon

Platoon level is responsible for the manoeuvres which lead to entry, exit, changes of lane, and the formation and dissipation of platoons. Most of it is vehicle-borne. There is a platoon-level
controller on each vehicle which oversees the actions of lower-level controllers of vehicle motion in such a way as to execute platoon manoeuvres safely and successfully at the right time and place. We will describe its operation later (see Section 8.3).

There is also a vehicle state vector (VSV). The VSV is a software record containing the AL license, the route and a number of variables describing the current position of the vehicle and what it is doing. The VSV is held in an asynchronous record (AR). An AR is a computer storage device which can be written to and read from by several different computers which are not in sync. The VSV, in fact, can be accessed by all the controllers on the vehicle. It can also be accessed, via the communication devices, by other vehicles and the various roadside controllers. Among other fields, it contains a “busy” flag. This is set during any manoeuvre, and will usually preclude any other manoeuvre.

The fields in the VSV are these below:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID#</td>
<td>vehicle identity</td>
</tr>
<tr>
<td>ln#</td>
<td>lane number</td>
</tr>
<tr>
<td>optsize</td>
<td>platoon optimum size</td>
</tr>
<tr>
<td>pltn#</td>
<td>platoon number</td>
</tr>
<tr>
<td>pos</td>
<td>position in platoon</td>
</tr>
<tr>
<td>busy</td>
<td>busy marker (see above)</td>
</tr>
<tr>
<td>sectlen</td>
<td>section length</td>
</tr>
<tr>
<td>fault</td>
<td>identity of any fault (see below)</td>
</tr>
<tr>
<td>mode</td>
<td>operating mode (see below) of current lane</td>
</tr>
<tr>
<td>xspeed</td>
<td>optimum speed in lane to right</td>
</tr>
<tr>
<td>hwy#</td>
<td>highway identity</td>
</tr>
<tr>
<td>sect#</td>
<td>section number</td>
</tr>
<tr>
<td>optspeed</td>
<td>platoon optimum speed</td>
</tr>
<tr>
<td>ownsize</td>
<td>size of own platoon</td>
</tr>
<tr>
<td>loc</td>
<td>odometer reading (see Section 4.5)</td>
</tr>
<tr>
<td>validation</td>
<td>valid AL license</td>
</tr>
<tr>
<td>speed</td>
<td>actual speed of vehicle</td>
</tr>
<tr>
<td>maxspd</td>
<td>maximum speed on freeway</td>
</tr>
<tr>
<td>safsp</td>
<td>minimum platoon separation at maxspeed</td>
</tr>
</tbody>
</table>
The state variables \texttt{maxspd} and \texttt{safsp} are set by the system on entry and are system-wide constants. They may vary from time to time, with the weather. There is a formula contained inside every vehicle which enables the safe interplatoon spacing to be calculated from these variables at any speed. \texttt{Optsize} is two numbers specifying a range of sizes within which platoons will neither merge or split. Like \texttt{optspeed}, it is set on entry by link and updated on each lane change.

At the roadside, there is a platoon-level controller associated with each block of freeway. A block is a length of some 1-5 miles of freeway. It may be convenient to have one entrance and one exit point per block: the point is of no great importance. There is a platoon-level controller at each point where a lane-change can occur. These controllers are hard-wired to the other platoon-level controllers in the same block. They can also communicate with vehicles in their immediate neighbourhood. Their function is more fully discussed later (see Section 8.5).

4.5 Architecture, Level 1: Regulation

The regulatory layer controls the movements of individual vehicles in response to signals from the platoon level and sensor readings from the physical level. It is entirely vehicle-borne. There are many components:

a. \textbf{Longitudinal Control} System. This will maintain a vehicle in platoon. If the vehicle is a platoon leader or a free agent, the control system will maintain the vehicle at platoon spacing from the next vehicle ahead in the same lane. Alternatively, the system may be falling back from the vehicle ahead or joining up to it when some special manoeuvre is being carried out.

b. \textbf{Lateral Control System}. This will maintain a vehicle on track in lane. It will also change lane at appropriate locations when requested to do so.

c. \textbf{Odometer}. As will be seen, at the entry to every block there is a block entry marker (b.e.m.). The odometer measures the distance of the vehicle from the b.e.m. and records this number in the VSV. The lane # is also recorded there. We call this data Zoc, the vehicle’s location. On entry, a loc is input by the system.

d. \textbf{Forward Sensor}. This instrument measures the distance of the vehicle from the one ahead. It also measures relative velocity directly. The readings are stored in an AR. It will operate at close range, in platoon. At longer ranges, it can keep track of which of several vehicles ahead is in the same lane. It will always detect vehicles up to a range exceeding the maximum value of platoon spacing.
e. Communication Devices. The description will imply that there are eight independent devices here. If any pair of the functions is lost, a hazard arises. It is possible to combine the functions in various ways. This will reduce the number of separate pieces of equipment. However, each function must then be carried out by more than one device. It is simpler, in specification, to speak of independent devices for independent functions. The devices are:

- Forward transmitter and receiver. In platoon, the devices can distinguish signals from the vehicle ahead from all others. Otherwise, their range is at least equal to sensor range. However, at a distance, there may be several vehicles with which communication is possible.

- Rearward transmitter and receiver. These have the same specification as the forward ones, except that they react to vehicles behind.

- Lateral transmitter and receiver. These communicate with vehicles in lanes on either side. Their range is such that they can reach vehicles adjacent to those in contact with those reached by the forward and rearward communicators.

- System transmitter and receiver. These communicate with the platoon-level roadside controllers.

- Message stores and flags. There are **ARs** for storing messages and flags indicating their presence. When a message is received, a flag is set in an **AR**. The message itself is stored in another **AR**. If the message is received from another vehicle in the same platoon, it is also passed to the appropriate transmitter. Thus a message is passed along the length of the platoon by the regulatory level controllers. There are two exceptions to this. A message clearly cannot be passed on by the first and last vehicles of a platoon. Also, there is a message type called “acknowledge-control” (see Section 8.2.2) which is not passed on. A message containing control data is also passed to the longitudinal controller.

f. Self-monitor. This will be called MON. MON checks the behaviour of the other vehicle-borne controllers and devices. If any fail, an appropriate fault marker is set in the VSV. The transmitters and receivers are “looping” at all times. Thus faults here are detected at once without special action at platoon level. As a vehicle passes the places where change of lane can occur, it receives a message which, among other things gives the current **loc**. This enables a check to be made on the odometer. The forward sensor
is checked at these points also. Here a special platoon-level-controller routine is provided.

4.6 Architecture, Level 0: Physical

The physical level, as the name indicates, is the one at which physical controls operate. An example is the way in which movements of the steering axle affect the vehicle heading. Another is the way reflections from other vehicles are interpreted by the forward sensor. In this paper these operations will be taken for granted.

5. PHYSICAL LAYOUT

Along the center of every AL there is a lateral reference. This is read by the vehicles’ lateral control systems and serves to keep vehicles on track.

As explained in Hitchcock, 1991, the hazards constrain the physical layout. In an automated lane, flows of 6000 vehicles/hour or more are envisaged. Maximum flow on a manual lane is around 2000 vehicles/hour. At such high densities, we must clearly admit the possibility that one vehicle will lose speed and be rear-ended. Operation in platoons is chosen for the design because, then, if there is a such a rear-end collision it occurs at low relative speed. We call this a follower’s collision. Follower’s collisions were first discussed by Shladover (1979). In platoon, a follower’s collision may be followed by a platoon crush, in which several vehicles successively rear-end one another at low relative speed. The resulting mass of vehicles will contain uninjured people, but will be moving at 50 mph or so. All will be well, provided there are no further collisions with platoons or the infrastructure. The following platoons in the same lane are separated by platoon spacing and so can brake to rest without collision. If, however, the platoon crush strays into an adjacent lane, it is very likely to hit some fixed bit of infrastructure, provided it is not hit by another platoon first. To prevent these high-relative-speed collisions, we must design the system with barriers, called fences, between the ALs. There are also fences between the leftmost AL and any other lanes that may be on the freeway. In order to permit change of lane there are gaps in the fences. These are called gates.

It violates the safety criterion to permit a manually-controlled vehicle on the ALs. The driver cannot be compelled to keep platoon spacing from the vehicle ahead. Therefore vehicles must be admitted to the system under automatic control. They enter through a gate from the transition lane (TL). The TL has both manually and automatically controlled vehicles on it. However the TL exists only in the neighbourhood of the gates which permit entry and exit. We shall see later
that automatically controlled vehicles on the TL must assume manual control or be brought to rest in a dormitory off the TL.

At the entry and exit gates there are system transmitters and receivers which, stimulated by a message from the driver, switch manual control off and on. At the exit gate this message is duplicated through the lateral and system communicators in case of failure of one channel on the vehicle. At every gate, including the entrances, there are also communicators at both link and platoon levels. The link level communicators receive destination messages and update the routing.

The platoon level communicators at gates “overhear” messages between vehicles concerning lane changing. On the lane to which the transfer is made, there is, by the gate, a vehicle presence detector (VPD). On lane from which transfer is made there is an active turning point, which is controlled by the platoon-level infrastructure controller. The turning point, when active, gives some form of electromagnetic signal which defines a precise location. If the design admits two-way gates there are VPDs and turning points on both sides. The turning point is probably a succession of electromagnets of particular polarity, but the technology used for the turning points is not important here. If a vehicle is arranging to pass through a gate, the controller inspects the VPD. If the VPD reveals that passage will not create a sideswipe, the turning point is activated. The vehicle control system detects the turning point, and commences passage through the gate. If the turning point is not activated it will remain in its original lane.

6. SYSTEM MODES

In fault conditions, it may be necessary to close a lane. It may contain a stalled vehicle, or even a crash. Less serious faults may require less fully degraded conditions. Operation can then continue, in a less effective way, which still permits the freeway to be used. The whole length of a lane between two off-gates (a section) must be in the same mode. There are some rules determining the modes admissible in one section when a nearby section is in some degraded modes. Apart from these different sections can be in different modes. We distinguish the following system modes.

a. Normal (N). This is used at all times when no faults are present.

b. No-Entry (NE). Here speeds are the same as in normal mode, but the gate controllers will not activate turning points to permit entry. NE mode is used when the section contains a vehicle which cannot communicate laterally.
c. **Slow-Ahead (SA).** Here speeds are reduced, but there are no other restrictions. This is used when an adjacent lane is Closed-Ahead (see below). In this case, many vehicles are trying to enter this lane. It is also used in some other fault conditions; for example, if a vehicle which cannot maintain full speed is trying to enter the lane.

d. **Closed-Ahead (CA).** This applies when there are sections of the lane ahead which are closed. It also applies in some other fault conditions. Speed in a CA section is reduced. Adjacent lanes are in SA mode. Gate controllers deny entry to the lane. At each gate, link-level messages advise vehicles how far they can travel in this lane and indicate a route change. If there is only one exit remaining before the stopped section is reached, speed is further reduced. If need be, vehicles stop and line up at the last gate.

e. **Crushstop (C).** In an emergency the system will send a message placing a section or sections into this mode and identifying a loc. There is some stationary object at this loc which blocks the lane. (For example, debris from an accident may have passed a gate.) Except in one case, all vehicles in the section immediately brake at full platoon braking and remain at rest. The exception is that vehicles within platoon spacing of the identified loc apply maximum braking. This will normally lead to in-platoon collisions. These are accepted as a lesser evil. Speed is thus reduced to the greatest extent possible. Perhaps collision with the obstacle is avoided altogether. At least the impact is ameliorated. Entry is barred. Further action is determined by the system controllers. They are likely to start by entering Stop mode (see below).

f. **Stop (S).** This may be initiated in the same way as C mode, except that there is no maximum braking. Alternatively, it may be entered after C mode. In this mode the system controllers can order some otherwise forbidden manoeuvres. These manoeuvres will include instructing vehicles to back up on the AL, and to exit through on-gates. The objectives will be to get emergency vehicles to an accident or breakdown, to get unaffected vehicles on their way, and generally to clear up the consequences of some untoward event. Details of all this remain to be designed.

g. **Resume (R).** This is used to restart a section in S, SA or CA. Again, details remain to be designed. The resumption is seen as ordered by the system controllers advised by human observation.
Modes can be degraded by the system. Reversion to normal operation is seen as requiring direct orders by a human. An exception is made for NE mode. When the faulty vehicle leaves the section, the section reverts to N mode.

7. **FAULTY VEHICLES**

The fault element in the VSV is set as follows:

- **0.** This indicates no faults, and is the only value consistent with setting of the validation marker in the AL license.

- **1.** This indicates an unreliable forward sensor. The vehicle cannot tell how far it is from the vehicle ahead of it, whether that vehicle is in the same platoon or another.

- **2.** This indicates loss of the ability to transmit messages or data to a vehicle ahead.

- **3.** This indicates loss of the ability to receive messages or data from a vehicle ahead.

- **4.** This indicates loss of the ability to transmit messages or data to a vehicle behind.

- **5.** This indicates loss of the ability to receive messages from vehicles behind.

- **6.** This indicates loss of ability to transmit to vehicles in adjacent lanes.

- **7.** This indicates loss of ability to receive from vehicles in adjacent lanes.

- **8.** This indicates loss of ability to transmit to the system.

- **9.** This indicates loss of ability to receive from the system.

- **10.** This indicates loss of ability to locate accurately.

- **11.** Any other fault (e.g. inability to maintain speed, inability to remain on track).

If a vehicle develops two or more faults, a hazard can arise at once. The vehicle is immediately brought to rest, and Stop mode is called. The design is such, however, that if one fault only is present, the vehicle can proceed without hazard, provided no other fault interacts. The other
fault may be on another vehicle. Therefore, the controller is such that if a fault flag is set the vehicle will immediately start a series of manoeuvres to quit the ALs. It may be possible for it to be driven normally on ordinary roads, so the driver need not be stranded. Further, a message, identifying the faulty vehicle is sent to the system, which keeps a list of faulty vehicles. They might decide to continue, to enter a degraded mode or to call for Stop mode. On leaving, the faulty vehicle advises the system that it has left. If the vehicle does not exit in a reasonable time the system controllers are advised, for in this position, human intervention is called for. The same is true if the system detects two faulty vehicles which are in danger of interaction.

8. OPERATION OF THE SYSTEM

We shall now describe how the system behaves as a whole, and then go on to describe how the platoon-level controllers achieve this. In the appendixes, detailed specifications of the modules making up the controllers are presented. System operations are divided into manœuvres and probes. There are some minor other activities associated with entry and exit. The probes are concerned to discover fault conditions by using equipment which is not otherwise engaged in such a way that non-functioning can be detected. We will discuss the manoeuvres first.

There are five basic manoeuvres. Only three, however, apply in normal operation. These are, in inverse order of priority:

a. Merge. A platoon of less than optimal size joins on to another.

b. Split. A platoon divides into two, separated by platoon spacing.

c. Change-lane. A free agent (only) transfers through a gate into an adjacent AL. Entry and exit are basically change-lane manoeuvres. The process is slightly different. Control has to pass from manual to automatic (or the reverse) at the right time.

The other two manoeuvres are initiated when there is a faulty vehicle.

d. Emergency-Change. A faulty free agent moves from one lane to another on its way out.

e. Forced-Split. A platoon divides in such a way that a faulty vehicle is either at one end of the platoon or becomes a free agent. One of the probes provides
means by which a non-faulty vehicle in a platoon can tell that one of its neighbours has faulty communication equipment, and therefore cannot advise it directly.

The three normal-operation manoeuvres are sufficient to enable a vehicle to do what is needed, provided it does not become faulty. It first enters the system, declares a destination, and receives a route. The route is updated every time it passes a gate. On its way through the system it will join with other vehicles to form a platoon when and if this is appropriate. When the vehicle’s route requires it to change lanes or exit, split manoeuvres enable it to become a free agent. Change-lane makes the lane change or exit. Hsu and her colleagues (Hsu, et al., 1991) showed rigorously that these three manoeuvres are sufficient to meet the requirement specification if no faults arise. A similar proof that the addition of the two extra manoeuvre protocols similarly conform to an appropriately revised requirement specification would require further research.

We shall now describe the manoeuvres, the probes, and the way the vehicle-borne controller works. However, full detail will not be given here. The usual procedure will be described. An indication will also be given of what happens if the usual procedure is inappropriate. The appendixes contain formal specifications of each module. First, we shall explain the purpose of the “busy” flag. Platoon manoeuvres are controlled, in the main, by platoon leaders. Usually, a leader will engage in only one manoeuvre at once. When the manoeuvre starts, the “busy” flag is set, and a message “request-[manoeuvre]” is sent, initiating it. At the end of the manoeuvre, the “busy” flag is reset. If the message initiating a manoeuvre is received by a second vehicle, the latter’s controller will consult the “busy” flag. If it is already set, the message “nack-request-[manoeuvre]” is sent. The “nack-request-[manoeuvre]” message acknowledges receipt of the message, but indicates inability to join in. This usually means that the would-be manoeuvrer must await completion of the existing activity. Then the request can be sent again. There is an exception to this. If a forced-split manoeuvre is requested, the existing manoeuvre, if there is one, will be suspended. This is because the forced-split request indicates the presence of a faulty vehicle in the platoon. For this reason, it may not be possible for the first manoeuvre to be completed normally.

8.1 Manoeuvres

8.1.1 Merge

Figure 2 shows a flow diagram for the merge manoeuvre. The merge manoeuvre is initiated by the leader of a platoon which is following another at a little over platoon spacing. Merge will be initiated when the following conditions are met:
Figure 2. Merge (after Hsu, et al., 1991).
a. The leader is not otherwise busy. Its busy flag is not set.

b. The route does not call for any manoeuvre.

c. There is no fault flag newly set, nor is the “busy” flag set.

d. There is a platoon ahead in the same lane.

e. The platoon is smaller than link control’s minimum.

The sequence of events needs little description. In Figure 2, A is the leader of the leading platoon. B is the initiator, the leader of the following platoon. If A is busy, or a merged platoon would exceed the optimum size, A sends the message "nack_request_merge." Otherwise, A sends "ack_request_merge." On receipt of this B increases speed until it is a metre or so behind C, the final vehicle of A’s platoon. It then sends “confirm-merge” which is passed up the platoon to A. A responds by updating its state vector. New control data are passed the length of the new platoon. Merge is complete.

Figure 2 does not show what happens if no reply is received to "ack_request_merge." In fact a message which indicates a potential fault somewhere is sent to the system. If many such messages are received, action will be taken to resolve the problem. Figure 2 does not indicate, either, that C also sets its “busy” flag during the manoeuvre. This has no effect unless a forced-split is called in A’s platoon during the manoeuvre. Full details are in the appendixes.

8.1.2 Split

Figure 3 shows the flow diagram for the split manoeuvre. Split is usually initiated by a member of a platoon that, because its route demands it, wishes to become a free agent and change lanes. It can also be initiated by the platoon leader if the size of the platoon exceeds the optimum. This will not happen as a result of a normal merge, but it can happen if a vehicle joins a platoon following a lane change.

Part of the change-lane protocol is a manoeuvre called split-change-lane. This makes a gap in an existing platoon. When the gap reaches a gate a vehicle can change lane into the gap. Except for the manner of its initiation, split-change-lane is identical with the split manoeuvre.

Three vehicles become busy during a split. In Figure 3, A is the original platoon leader. B is the vehicle which will become leader of the new platoon. C, not shown in Figure 3, is the vehicle immediately ahead of B. If A is immediately ahead of B, there is no C. There are two
Figure 3. Split (after Hsu, et al., 1991).
variants. If A wishes to become a free agent, it sends “invite-split” to B, which is then the second vehicle in the platoon. On receipt of “invite-split” or "ack_request_split" B sets new control data, which are passed to vehicles behind it. B, and the following vehicles, decelerate until B is platoon spacing behind C. It then sends “confirm-split.” Vehicles reset “busy.” The split manoeuvre is complete.

Not shown in Figure 3 are the messages sent to the system if no reply is received to these messages. Again, the “busy” flag for C is only effective if a forced-split occurs between A and C while the split is in progress. The appendix gives details.

8.1.3 Change-Lane

Figure 4 is the flow-diagram for change-lane. Here, again, three vehicles may be involved. A is the free agent wishing to change lanes. B is, if it exists, a platoon leader or free agent in the lane into which the change is to be made. C is a platoon leader or free agent in the lane beyond B. C becomes involved only if B does not exist. A starts the procedure with the message “request-change-lane” giving its own loc, speed, and the direction of the intended lane-change. Nearby platoon leaders in the two adjacent lanes on the appropriate side of A’s lane respond with an "ack_request_change_lane." If they are busy they respond with a "nack_request_change_lane." These messages include the locs of the senders. A now works out with whom it has to cooperate. A message "thanx_but_no" is sent to all those who sent "ack_request_change_lane," but need not become part of the manoeuvre. These may be vehicles in the lane two over (i.e., potential C’s) when there is a valid response from the closer lane (i.e., a potential B). Alternatively they may be too far away from A.

If there is a B, B itself will work out the best strategy. There are three variants. First, B may request A to decelerate (message “request-decelerate”). This message refers to a gate and a time when its final member has just passed the gate. A decelerates appropriately, and as it reaches the gate it sends "confirm_dropt." The gate also receives this message and, provided the receiving side is unoccupied, A will be able to change lane. A does so, and sends “confirm-change-lane." This initiates a merge procedure, as already discussed. When the merge is complete, change-lane is also complete.

Second, B may send “request-split-change-lane” to one of the members of its own platoon. The split takes place as already discussed. The message “confirm-split” will be received by B in due course. As B reaches the gate, it sends "confirm_split_change_lane" to A. This message includes a gate and a time at which the change will be made. Meanwhile, A has been maintaining speed, and should therefore reach the specified gate at or very near the stated time. The gate reacts to the message also. A changes lane. A’s message “confirm-change lane” starts a merge procedure, as already discussed.
Figure 4. Change-Lane
(Change is to lane 1: platoon-leader B is in lane 1.)
lane 2 is adjacent to lane 1: leader C is in lane 2.)

20
Finally B may itself decelerate. Having done so, B sends “confirm-decelerate” as it reaches the gate. Again, this message names a gate and a time, which will be just ahead of B. Again A’s message “confirm-change-lane” starts a merge procedure as already discussed. Change-lane is complete.

If there is a C, all that is required of it is that it hold its relative position to A. In particular it will not change lane itself. Furthermore, it occupies the position where a vehicle in its lane would have to be for a change-lane to interfere with the changer. A will send the message “change-to-void” which names the gate through which A will change lane, as well as the time of the change. The specified gate will activate the turning point, provided there is no obstacle to doing so. A will pass through the gate, and will send “confirm-change-lane.” Both A and C will reset “busy.” Change-lane is complete.

If there is neither a B nor a C, A must still send “change-to-void” to activate the gate. A changes lane and the manoeuvre is complete.

Not shown in Figure 4 are the actions taken if no reply is received after some time. Under these conditions a message is sent to the system, and the manoeuvre is canceled. It may be restarted later. If, when it has changed lane, A finds itself too far away from the platoon with which it is trying to merge, a message is again sent to the system. The most likely reason for this is that one vehicle has developed a fault in the odometer. Finally, if during the earlier stages of the manoeuvre forced-split is called, change-lane is called off. If however, the change of lane is already committed, the following merge continues. All these extra details are discussed in the appendix.

8.1.4 Emergency-Change

Figure 5 is the flow diagram for emergency change. Emergency-change is initiated by a faulty vehicle which has become a free agent. Such a vehicle is now programmed to leave the ALs as soon as possible. Because the vehicle is faulty, the usual change-lane procedure is inappropriate. Change-lane requires several subsystems, any of which may be the site of the fault. Change-lane involves close approach to other vehicles and formation into new platoons, which is also inappropriate when a vehicle is faulty.

Emergency-change involves cooperation with four other platoon leaders. These are requested to run a platoon spacing ahead, and a platoon spacing behind the faulty vehicle. Two of the four are in the lane into which the change will be made. The other two are in the lane beyond. All simply maintain themselves in this position while the change is made into a large gap. The four shepherding vehicles adopt the speed, whatever it may be, of the faulty vehicle.
Figure 5. Emergency Change-Lane.
(Change is to lane 1: B and C in lane 1, B leading)
(lane 2 is adjacent to lane 1: D E in lane 2, D leading)
SA mode = Slow-Ahead, NE = No-Entry
vehicle has changed lane it sends “confirm-emergency-change. ” Busy flags are reset. The shepherds are free to engage in other manoeuvres.

There are a number of special features, activated in particular circumstances. The fault may be in a lateral communicator (fault 6 or 7). If so, the lane in which it is running is in NE mode. When the change is made, the clear lane reverts to N mode. The one which now contains the faulty vehicle is put to NE. Another feature which comes into play when fault 6 or 7 is present is that the system communicators come into play. Messages which cannot be transmitted or received by the lateral communicators are relayed through the system communicators. As will be seen, the gate controllers communicate with both the lateral and system communicators, so that if a vehicle is faulty in either the lateral or the system communicators the non-faulty one of the pair passes the message to activate the turning point.

If forced-split is called in either shepherding platoon, the manoeuvre is broken off. We now have two faults in different vehicles interacting. The system controllers are advised. This is an area where hazards can be readily generated and human supervision is therefore necessary. Full details are given in the appendixes.

8.1.5 Forced-Split

The forced-split manoeuvre is initiated by a vehicle in a platoon which develops a fault. A faulty vehicle has to quit the ALs. The first step is to become a free agent. Usually, the reaction is to call for a split ahead so that the faulty vehicle becomes platoon leader. If, however, the fault is in the rearward communicators, the initial split is behind the faulty vehicle. If the faulty vehicle is platoon leader, the split is also behind it. We shall see later that even if the fault prevents full communication with other vehicles in the platoon, the probes enable the other vehicles to operate the forced-split.

Forced-split differs from split in these ways:

a. A confirm-forced-split message is sent when the manoeuvre is complete. However, no reliance is placed on its being received. After a reasonable interval of time, the manoeuvre is terminated.

b. The original platoon may send a message breaking off some other activity; if not, this leader checks, when the forced-split is over, to see if another activity should be resumed.

c. The last vehicle in the rear platoon will receive data indicating that it is a member of a new platoon. It consults it own “busy” flag. If the flag is set, another
manoeuvre is in progress. A message is sent to the new platoon leader indicating as much.

Because these differences do not show up well in a flow diagram, none is shown for this case. Figure 3 applies here too.

**Forced-split differs** from other manoeuvres in that it overrides the “busy” flag. A faulty vehicle in a platoon is either an imminent danger causing a crash (probably a low speed one) or interferes with the execution of the manoeuvre. Therefore the manoeuvre is preempted.

In the case of a split or merge manoeuvre, the manoeuvre continues in the unaffected platoon. The leaders in a platoon which is engaged in a forced-split manoeuvre can receive messages. If they do, however, the message is stored and is not responded to until the forced-split is complete. Then the system recovers stored messages and either sends the "confirm_[manoeuvre]" message or continues with a merge or split. If the forced-split occurred in the first of the two Platoons engaged in a merge or split, the original leader of this platoon is now inaccessible. The last member, however, does have “busy” set, and has stored the necessary data about the previous manoeuvre. This last member sends this forward to the new leader. An exception arises when the faulty vehicle is itself the last member of the platoon. Now a merge manoeuvre will be terminated. A message is passed to the following platoon to fall back at the time the forced-split is initiated.

If forced-split is called in a cooperating platoon, a change-lane manoeuvre is terminated unless the change of lane has already taken place, or the gate has been activated for the change. At this stage in a change-lane manoeuvre, a merge is in progress or about to begin. It will continue as other merges do.

Further detail about this manoeuvre is given in the appendix. Formal specifications of all modules are included.

### 8.2 Probes

#### 8.2.1 Platoon Leader Probe

Figure 6 gives a flow diagram for the platoon leader probe. The platoon leader probe is a check on the continued function of the forward sensor. The probe is initiated whenever a platoon leader reaches a gate, unless it is “busy.” If the forward sensor perceives at least one vehicle ahead, a message is sent giving the loc of the sender and indicating the distance of the vehicle perceived. A vehicle receiving this message may reply if it is the last vehicle in a platoon. There may be many such vehicles. Each will reply if its own loc makes it likely that it is the
Figure 6. Platoon Leader Probe.
vehicle addressed. If the sender receives a reply from a vehicle whose loc makes it likely that it is the one seen, all is well.

If, however, no reply is received by the first sender, another message is broadcast. This message calls for a reply from any vehicle credibly being addressed. This is because, if the road is strongly curved, it may be difficult to be precise about the relative position of locs on diverse lanes. If, again, a reply is received and appears to be valid, all is well. Otherwise there are two possibilities. Either the sender is seeing things which are not there, or the receiver, is unable to communicate. If there is or seems to be more than one potential receiver, the first alternative is valid. Fault #1 is set by the probing vehicle and the system is advised. If there is or seems to be just one potential receiver, a message is sent to the system. The system may reply that indeed, there is a vehicle in this general area with such a fault. If so, no further action is taken. Otherwise fault #1 is set by the prober. The other possibility is that no vehicle is perceived, in which case a message is broadcast to any potential receiver ahead saying so. If a reply is received, one tries again. If again a reply is received, fault #1 is set. The system is informed.

8.2.2 In-Platoon Probe

Figure 7 is a flow diagram of the in-platoon probe. If a vehicle in a platoon develops a communication fault, it will set a fault flag. From its side, the faulty vehicle will start a forced-split. However, if the fault is in the forward or rearward communication system, the vehicle with which communication has failed may not be aware of the need to forced-split. The in-platoon probe deals with this. Every time a message with control data is sent with the leader’s VSV data down the platoon, each vehicle responds ahead with an “acknowledge-control” message. As explained in Section 4.5, this message, uniquely, is not passed on up the platoon. It is recognized that a number of temporary conditions will interfere with the passage of one message. Two consecutive failures to receive a control message are required before the vehicle deduces that there is a failure ahead of it. It then sends a temporary leader’s message, giving its own control data. The would-be leader may subsequently receive a message from another temporary leader ahead of it. This can occur when the break in communication is not immediately ahead of it. If a temporary leader receives a message from ahead, it adopts a follower’s role again. If not, it will send a message in both directions, indicating that a forced-split will be formed ahead of it.

Alternatively, there may be no acknowledgment. Once again, confirmation that the loss is permanent is awaited. After that, a message is sent, both ways, indicating that a forced-split will occur behind the vehicle sending the message.
Figure 7. In-Platoon Probe.
(The probe operates cyclically: strictly, there is no entry point. Comprehension is eased by starting at the point indicated.)
8.3 Entry and Exit

Entry has two components. First the vehicle must signal the system that it wishes to join the system, and give a destination. The system will reject the vehicle’s request for entry if it does not signal that its AL license is valid. It will also reject the request if the destination is inaccessible. This could happen if the driver is trying the wrong direction, or if there is some breakdown ahead.

If a vehicle is accepted, the next stage is a change-lane. This differs in two ways from other change-lane manoeuvres. The driver, before requesting entry, can match speed and position so as to make entry simpler. Also, should entry prove to be impossible for any reason, the vehicle will ultimately pass the last on-gate. It will now be treated as if it had just exited. The driver will be invited to resume manual control.

Exit, too, starts with a change-lane, like any other, on to the TL. However, in this case, the system sends a warning to be ready to resume manual control on exit. Immediately after exit is confirmed by the VPD, the driver is offered the opportunity to resume manual control. If manual control is not taken, the vehicle will be first slowed down as it passes a particular communicator near the end of the TL. If it reaches the end of the TL, it will be brought to rest in a dormitory.

8.4 Operation of the Vehicle-Borne Controller

The controller has many functions, and cannot remain with any operation for a significant period. Instead it cycles, continually, through six functions. If any of these is active, the controller will enter it. The controller will initiate some operation, if it is time to do so. Otherwise, the controller will simply set a flag. When control returns on the next cycle, the controller will pick up the operation where it left off. In the interval it has examined the need for operation in the other five categories.

All this is set out in detail in the appendixes. Figure A.1 shows a schematic, naming the modules. A full description of the latter will be found in the appendix “Supervisor.” The six elements operate as described below. We have given the elements mnemonic names.

a. **Buschek.** “Buschek” is “check busy.” If the busy flag is reset, control passes on. Otherwise the controller consults flags to discover which manoeuvre is in progress. Another flag indicates the role played by this vehicle, while a third records the stage reached.
b. **Forprob.** “Forprob” is “Forward probe.” This section progresses the platoon leader’s probe. Again, flags are set, indicating that the probe is in progress. Another flag advises how far the probe has progressed.

c. **Messrec.** “Messrec” is “Messages received” This module is responsible for initiating new actions in response to messages. Flags have been set in the message register by the regulatory level communicator controllers indicating which messages have been received. This level will also already have passed on many up or down platoon. Regulation level will also have discarded messages which do not apply to this vehicle. Other messages are of concern to particular elements, and their flags are noted there.

However, some messages are initiators. “Request_merge” is an example. Another is “Probe-l,” the first forward probe message from a **vehicle** to the rear. “Messrec” notes these. It examines each incoming message in turn, and transfers control to appropriate elements.

d. **Newfalt.** “Newfalt” is “New fault.” Fault flags are inspected to see if any are set. The fault-l probe is called from here. This probe is used by a vehicle which has lost forward sensor capacity, and has no vehicle ahead of it in platoon. It calls forward giving its **loc**. Vehicles ahead in the same lane give their own **locs**. These data are passed to the longitudinal control system instead of the sensor readings. Here too, messages are sent if a vehicle is for some reason moving very slowly.

e. **Actcalled.** “Actcalled” is “Calls for Action.” If faults are present, the appropriate manoeuvre is called. If there are none, the module examines the routing to see if any manoeuvre needs to be initiated. If the platoon is above or below the optimum range, it may be appropriate to send a “request-merge” or "request_split." At a gate, it may be appropriate to initiate the platoon-leader’s probe.

f. **Contdat.** “Contdat” is “Control data.” The control data are passed along the platoon by the regulation level which also makes them accessible to longitudinal control. However, the in-platoon probe also needs these data, and the logic of this probe is carried out in Contdat.

This completes the general description of the system. For further details the appendix should be consulted.
8.5 Operation of the Roadside Controller

In fault-free operation, the only activities of the roadside controllers are to send signals causing manual and automatic control to be switched, and to activate the turning points at gates. In fault conditions, the system communicators also act as a backup to the normal ones. The gates communicate both with the lateral and with the system communication subsystems. The system is called on, whenever there may be a vehicle that cannot communicate over a distance. (The description of the platoon leader’s probe in Section 8.2.1 contains an example.) Also, the system maintains two lists of faults and potential faults. Whenever a fault flag is set, the roadside controller is advised. When a faulty vehicle exits it is removed from the list. The system controllers are advised if there is too long an interval before a faulty vehicle exits. These people also receive a message if there are several faulty vehicles close together.

Further, there are a number of events which are reported to the system which indicate that something is faulty somewhere, but do not enable a culprit to be identified. Excessive delays or failures to respond in a merge process are examples. Another example is a vehicle changing lane too close to or too far away from a partner in the change or emergency-change protocol. Such events result in vehicles being put on a potential-fault list. If they appear several times, a message is sent declaring them faulty.


It has already been explained (see Section 1) that this specification is the completion of one due to Hsu and her collaborators. These authors have agreed to their work being used in this way. While their work is not a large part of what is described here, it did, of course, inspire the whole. In particular Hsu and her colleagues did not specify a physical layout. They did not describe the cyclic action of the vehicle-borne controller. Two of the manoeuvres (forced-split and emergency-change) have been added. So have both the probes. The whole of the treatment of non-reply to messages described here is new.

However, the small fraction of the pages do contain a vision of how a system might work that permeates the whole of this report. This work does not address the same topics as Hsu, et al. Its conclusions are, it is believed, useful and are not trivial consequences of the previous work. Nevertheless, this work has been greatly aided by being able to build on the earlier work.
ACKNOWLEDGEMENTS

The author wishes to acknowledge the technical support and encouragement of Dr. Steven E. Shladover, Technical Director, PATH. The author experienced some difficulty in writing this paper as clearly understood prose. The comments of Mr. Sompol Chatusripitak of Caltrans were very helpful.

The great value of the work of the authors of Hsu, et al. (1991) has already been acknowledged.
REFERENCES


APPENDIXES

A.1 INTRODUCTION

The appendixes contain formal statements of the form and effect of each module. The form selected is loosely based on the statements required in a number of formal computer languages. There is, however, no computer language known which can accept these statements. Since many of the terms used are not axiomatized, a great deal more would be needed before these formal statements could be so used. However, this form has been found to be adequate for manual verification of the statements made in the course of a fault tree analysis, and this is the application intended.

The general section of these appendixes contains the modules which refer to more than one vehicle or platoon. The others are grouped into sections. Each section refers to the supervisor itself (Figure A. 1) or to one of its six elements. However, the figures for the element Busycheck are broken down into the five manoeuvres (Figures A.2 - A.6). At the end, the “System” section describes the operation of the platoon-level roadside controller.

The following abbreviations are used for the system modes:

\[
\begin{align*}
N & \rightarrow \text{Normal} \\
CA & \rightarrow \text{Closed-Ahead} \\
R & \rightarrow \text{Resume} \\
NE & \rightarrow \text{No Entry} \\
SA & \rightarrow \text{Slow-Ahead} \\
S & \rightarrow \text{stop} \\
C & \rightarrow \text{Crashstop}
\end{align*}
\]

A.2 HOW TO USE THE MODULE DEFINITIONS

If it is wished to determine exactly how the system, or some part of it, works, it is necessary to consult the written module definitions. The figures provided here do act as a guide to the flow of control through the modules. However, they are not a full flow-chart. Where a module is at a branch, for example, the figures do show two successors. They do not say what the branching criterion is. That is done in the written module definitions. Similarly, a branching criterion may require external input. The figure may show that there is a predecessor which provides the relevant data — but the message itself is only identified in the text. Equally the words within a box in a figure are necessarily abbreviated, and often omit relevant information.

In general, as each of the elements in the main routine (called supervisor) is entered, the first action is to consult flags. The flags indicate whether an operation is in progress, and if so, the point which it has reached. Thus, for example, if Busycheck is entered, the system first checks the busy flag. If the flag is set, a manoeuvre is in progress. Alternatively, the flag may just have been set by Actcalled. In either event, the system first determines the role the vehicle is playing. Flags with names like split(2) or emerch(3) indicate the vehicle’s role. In fact a vehicle with split(2) set is the leader of the second platoon in a split manoeuvre. Emerch(3) indicates the leader of the trailing platoon in the lane which is to be entered in an emergency change. Within this role, there may be stages. Thus with split&) set, and the stage flag = 3, the newleader is...
dropping back to platoon spacing. The action is to consult the reading of the forward sensor, and determine if the full separation is yet achieved.

So much is perhaps reasonably clear from the diagrams. But for full understanding, the text must be used. An example may be useful. Suppose that a platoon is alone on the road except for a vehicle ahead in the same lane which has fault #5. The vehicle thus cannot receive a message from the leader of our platoon. A gate is reached. When the element Actcalled is reached, the forward probe is started. To see how consult Figure A.10. In the circumstances described Actcalled will successively call the modules Linkmess, Mersplit and Startprobe. By looking at each specification it is apparent that Startprobe will set the stage flag in Forprobe = 1.

Now examine Figure A.7. It is perhaps unnecessary to examine the specifications of the modules Forprobe, Insight, Recackprobel, Probeagain, and Recackprobe2. On successive calls to Forprobe, as the specifications will confirm when examined, the flag advances from 1 to 8, Probe-1 is sent (flag = 1) and later Probe-2 (flag = 5) is sent. Since the vehicle addressed does not receive the message there is no reply. At flag = 8, Recackprobe2a is called. No message has been received. However there is only one vehicle in sight. So the flag is set to 16, and the message Fault_1_veh is sent to the system. Now consult the system section. The system will consult its list of faulty vehicles. It will discover that there is indeed a vehicle with fault 4 or 5 in the area from which the message is sent. So it sends No-fault-1. Whatever the relative cycles of system and the vehicle controller, there is sufficient time for this message to be picked up when the flag is 16, 17 or 18. Nofault then resets the flag in Forprob. No fault is wrongly ascribed to our platoon leader.
A.3 GENERAL

These modules are general. They are called from several locations, and have no specialised successors:

**Name:** Cacall  
**Admitted in:** N NE SA  
**Input:** Locations  
**Requires:** -  
**If branch:** Yes  
**Condition:** In range stated? Pos?  
**Effect:** 1. If within range stated, \( \text{maxspeed} \) to reduced speed; update state vector; if \( pos=1 \), full platoon braking;  
2. Endif; endif; return;

**Name:** Callstop  
**Specification in:** Emerch (6)  
**Fig:** -  
**Admitted in:** N NE SA CA  
**Input:** -  
**Requires:** Several precursors  
**If branch:** No  
**Condition:** -  
**Effect:** Message to system - systems sets crashstop mode from pos of Vehicle calling, back twice interplatoon spacing; sets CA mode behind that and SA mode in parallel lanes; inform system supervisors;

**Name:** Crashstopbehind  
**Admitted in:** N NE SA CA  
**Input:** -  
**Requires:** Several precursors  
**If branch:** No  
**Condition:** -  
**Effect:** Message to system - systems sets crashstop mode from pos of Vehicle calling, back twice interplatoon spacing; sets CA mode behind that and SA mode in parallel lanes; inform system supervisors;

**Name:** Crashstopcall  
**Admitted in:** N NE SA CA  
**Input:** Locations, and perhaps name of platoon  
**Requires:** -  
**If branch:** Yes  
**Condition:** Within platoon spacing of loc named? In range stated? Pos?  
**Effect:** 1. If in named platoon, or within platoon spacing behind loc stated,  
2. \( \text{Maxspeed} \) to zero; full braking;  
3. Endif; If within range stated, \( \text{maxspeed} \) to zero; if \( pos=1 \), full platoon braking;  
4. Endif; endif; return;
Name: Cutspeed
Admitted in: N NE SA CA
Input: Locations, speed
Requires: -
If branch: Yes
Condition: In range stated? Pos?
Effect:
1. If within range stated, $\text{maxspeed}$ to stated speed; store original $\text{maxspeed}$;
2. Endif; return;

Name: Maystop
Admitted in: N NE SA CA
Input: Locations
Requires: -
If branch: Yes
Condition: In range stated? Pos?
Effect:
1. If within range stated, set flag in Newfalt;
2. Endif; return;

Name: Necall
Admitted in: N NE SA
Input: Locations
Requires: -
If branch: Yes
Condition: In range stated? Pos?
Effect:
1. If within range stated, update state vector;
2. Endif; endif; return;

Name: Norcall
Admitted in: NE SA
Input: Locations
Requires: -
If branch: Yes
Condition: In range stated? Pos?
Effect:
1. If within range stated, $\text{maxspeed}$ to stored value; update state vector;
2. Endif; return;

Name: Scall
Specification in: Emerch (6)

Name: Sacall2
Admitted in: N NE
Input: Locations
Requires: -
If branch: Yes
Condition: In range stated? Pos?
Effect:
1. If within range stated, $\text{maxspeed}$ to reduced speed; store original $\text{maxspeed}$; update state vector; if pos = 1, full platoon braking;
2. Endif; endif; return;
Name: S topcall
Admitted in: N NE SA‘CA
Input: Locations
Requires: -
If branch: Yes
Condition: In range stated? Pos?
Effect: 1. If within range stated, maxspeed to zero; if pos = 1, full platoon braking;
2. Endif; endif; return;
Figure A. 1. Supervisor.
A.4 SUPERVISOR (Figure A.1)

Name: Actcalled
Admitted in: N NE SA CA
Input: Messages from link control
Requires: Newfalt or Takeover
If branch: Yes
Condition: Fault present? Message from link? Platoon too big, small? Time?
Effect: 1. If a fault flag is set, then if emergch(1) is set, or forspl(x), x = 1 - 5 is set,
2. Else send to system fault_#x_present with vector; if ownsize > 1, then if fault #4 or #5 is present,
3. Call Fault4or5;
4. Else if pos = 1, call Forspl1;
5. Else call Forsplal;
6. Endif; endif; else call Emerchcall;
7. Endif; endif; else call linkmess;
8. Endif; Go to contdat;

Name: Buschek
Admitted in: All
Input: -
Requires: Contdat (in iterating cycle)
If branch: Yes
Condition: Busy? Internal flags?
Effect: 1. If busy set, examine flags in order shown until one found set; then, if need be, examine flag2, and call routine shown in list;
2. Endif; goto Forprob;

<table>
<thead>
<tr>
<th>Flag1</th>
<th>Flag2</th>
<th>Routine to call</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chng(1)</td>
<td>1,2,3</td>
<td>Repwait1 4</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Voidch 4</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Recreqdecel 4</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Pass2 4</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Pass3 4</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Pass1 4</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>Sendconchll 4</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Sendconchll2 4</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>Sendconchll3 4</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>Sendcondropt 4</td>
</tr>
<tr>
<td>Chng(2)</td>
<td>1,2,3</td>
<td>Repwait2 4</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Compposns 4</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Reccondropt 4</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Sendcondecel 4</td>
</tr>
<tr>
<td>Flag 1</td>
<td>Flag 2</td>
<td>Routine to call</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Chng(2)</td>
<td>7</td>
<td>Reconcll2</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Reconcll1</td>
</tr>
<tr>
<td>Chng(3)</td>
<td>1,2,3,4,5</td>
<td>Reconcll3</td>
</tr>
<tr>
<td>Emerch(1)</td>
<td>6</td>
<td>Repwate</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Voidemerch</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Compposn</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>Recadjahead</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Passemerch</td>
</tr>
<tr>
<td>Emerch(2,3,4,5)</td>
<td>1,2,3,4,5</td>
<td>Nset3</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Repwate1</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Recposn</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Inposn</td>
</tr>
<tr>
<td>Forspl(1)</td>
<td>1</td>
<td>Reconemer</td>
</tr>
<tr>
<td>Forspl(2)</td>
<td>2</td>
<td>Qfaltl</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Newlead</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Sepmeasfsl</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Qslowmode</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Testactflag</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Splitagain</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Recfsplitover</td>
</tr>
<tr>
<td>Forspl(3)</td>
<td>1</td>
<td>Recconfspl2</td>
</tr>
<tr>
<td>Forspl(4)</td>
<td>2</td>
<td>Recconfspl3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Recconfspl4</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Recconfspl5</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Qmess</td>
</tr>
<tr>
<td>Forspl(6)</td>
<td>1</td>
<td>Recackreqm</td>
</tr>
<tr>
<td>Merge(1)</td>
<td>2</td>
<td>Recunmerge</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Norepm3</td>
</tr>
<tr>
<td>Merge(2)</td>
<td></td>
<td>Reconmer</td>
</tr>
<tr>
<td>Merge3</td>
<td></td>
<td>(No action)</td>
</tr>
<tr>
<td>Split(1)</td>
<td>1</td>
<td>Recconspl</td>
</tr>
<tr>
<td>Split(2)</td>
<td>2</td>
<td>Recackreqspl</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Newlead</td>
</tr>
<tr>
<td>Split(3)</td>
<td></td>
<td>Sepmeas-s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Recsplitover</td>
</tr>
</tbody>
</table>
Name: Contdat
Admitted in: N NE SA CA
Input: -
Requires: Actcalled
If branch: Yes
Condition: Flags 1, 2
Effect: 1. Reset control data flag (1), ack_cont flag (2) and message counters;
2. For each message recorded,
   a. If it is a special message then if it refers to this vehicle (see below),
      increment counter;
   b. else reset message flag;
   c. endif; endif; if it is a control data message, set flag 1;
   d. endif; if it is an ack_cont message, set flag 2;
   e. endif;
3. Call Controlmess;
4. Endif; go to Buschek;

Note. 1. At regulation level, action of in-platoon reception/transmission is to:
   a. examine nature of message received; set flags and counters (using same file
      as messrec) indicating which messages have been received.
   b. If it is a control data message, send data to controllers.
   c. If it an ack_cont_dat message do no more
   d. Otherwise, unless pos = 1 for a message from behind, or pos = last for a
      message from ahead, pass message on.

Note. 2. “Special” messages are:

<table>
<thead>
<tr>
<th>Message</th>
<th>Pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Going forward:</td>
<td></td>
</tr>
<tr>
<td>Request-merge</td>
<td>1</td>
</tr>
<tr>
<td>Request-split</td>
<td>1</td>
</tr>
<tr>
<td>For-split_b_N_f</td>
<td>1, N</td>
</tr>
<tr>
<td>Going backward:</td>
<td></td>
</tr>
<tr>
<td>Ack_request_merge</td>
<td>last</td>
</tr>
<tr>
<td>Request-mer_3_set</td>
<td>last</td>
</tr>
<tr>
<td>Ack_request_split_N</td>
<td>(N-1)</td>
</tr>
<tr>
<td>Invite-split-N</td>
<td>(N-1) if N &gt; 2, N</td>
</tr>
<tr>
<td>Req_spl_chInRange_N</td>
<td>(N-1), N</td>
</tr>
<tr>
<td>Forsplb_N_r</td>
<td>N+1, last</td>
</tr>
</tbody>
</table>
Name: Forprob
Admitted in: N NE SA CA
Input: -
Requires: Buschek
If branch: Yes
Condition: Flag?
Effect: 1. If flag set, call routine according to list below;
   2. Endif; go to Messrec;

<table>
<thead>
<tr>
<th>Flag</th>
<th>Routine</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Insight</td>
</tr>
<tr>
<td>2,3,4</td>
<td>Recackprobe 1</td>
</tr>
<tr>
<td>5</td>
<td>Probeagain</td>
</tr>
<tr>
<td>6,7</td>
<td>Recackprobe2</td>
</tr>
<tr>
<td>8</td>
<td>Recackprobe2a</td>
</tr>
<tr>
<td>9,10,11</td>
<td>Recackprobenov 1</td>
</tr>
<tr>
<td>12</td>
<td>Novehnorep</td>
</tr>
<tr>
<td>13,14,15</td>
<td>Recackprobenov2</td>
</tr>
<tr>
<td>16,17,18</td>
<td>Reclveh</td>
</tr>
<tr>
<td>19</td>
<td>Setfault# 1</td>
</tr>
<tr>
<td>20</td>
<td>Sendackprobel</td>
</tr>
<tr>
<td>21</td>
<td>Sendackprobe2</td>
</tr>
<tr>
<td>22</td>
<td>Sendackprobenov</td>
</tr>
</tbody>
</table>

Name: Messrec
Admitted in: N NE SA CA
Input: Message list (see below)
Requires: Forprob
If branch: Yes
Condition: Any messages received from other platoons or system?
Name: Messrec (continued)

Effect:

1. If busy,
2. Else examine existing flags; for each:
   a. If more than 5 cycles old, reset flag; decrement counter;
   b. endif;
3. Endif; if any messages, then for each:
   a. If emergency message from system, delete message; call appropriate
      routine; endif;
   b. If unaddressed initiating message (see below), check pos; if
      appropriate to this vehicle, call appropriate routine (see below);
   c. Delete message; decrement counter;
   d. Endif; endif; examine address, and message type; if to this platoon,
      and pos is appropriate, check for spl(x) in Buschek;
   e. If for spl(x) is set, store message;
   f. Else set message flag with cycle number;
   g. Endif; endif;
4. Endif; goto Newfault;

Notes. The action of a receiver is to store each message, as received, and increment a
message counter. The same list and counter is used by the control data receivers,
whose messages are also acted on here.

Emergency messages from system

These are obeyed by all vehicles:

Cstopcal
Stopcall
Cacal
Sacal
Necal
Cutspeed
Norcal
Name: Messrec (continued)

Unaddressed initiating messages

These are noted only by vehicles in appropriate pos:

<table>
<thead>
<tr>
<th>Message</th>
<th>Pos</th>
<th>Routine Called</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request-merge</td>
<td>1</td>
<td>Recreqmer</td>
</tr>
<tr>
<td>Request-merge</td>
<td>Last</td>
<td>Passon</td>
</tr>
<tr>
<td>Ack_request_merge</td>
<td>Last</td>
<td>Recackreqmerbeh</td>
</tr>
<tr>
<td>Request-mer_3_set</td>
<td>Last</td>
<td>Recmer3set</td>
</tr>
<tr>
<td>Ack_request_split_N</td>
<td>(N-1)</td>
<td>Recackreqsplitbeh</td>
</tr>
<tr>
<td>Request-split</td>
<td>1</td>
<td>Recreqsplit</td>
</tr>
<tr>
<td>Request_split_chnge_in_N</td>
<td>N</td>
<td>Recreqsplitbeh</td>
</tr>
<tr>
<td>Request_split_chnge_in_N-1</td>
<td>N-1</td>
<td>Recreqsplitbeh</td>
</tr>
<tr>
<td>Invite-split-N</td>
<td>N</td>
<td>Recinvsplit</td>
</tr>
<tr>
<td>Invite-split-N-1</td>
<td>N-1</td>
<td>Recackreqsplitbeh</td>
</tr>
<tr>
<td>Forspl_beh_N_f</td>
<td>1, (N &gt; 1)</td>
<td>Leadforsplit2</td>
</tr>
<tr>
<td>Forspl_beh_N_f</td>
<td>N</td>
<td>Recbehinda</td>
</tr>
<tr>
<td>Forspl_beh_N_r</td>
<td>N+1</td>
<td>Recaheadb</td>
</tr>
<tr>
<td>Forspl_beh_N_r</td>
<td>Last(&gt; N+ 1)</td>
<td>Lastbusy</td>
</tr>
<tr>
<td>Request-chng-In</td>
<td>1</td>
<td>Recreqchln</td>
</tr>
<tr>
<td>Request-emerch</td>
<td>1</td>
<td>Recreqemerch</td>
</tr>
<tr>
<td>Fault-lqrobes</td>
<td>last</td>
<td>Recfault lprobe</td>
</tr>
<tr>
<td>Probe-1</td>
<td>last</td>
<td>Recprobe 1</td>
</tr>
<tr>
<td>Probe-2</td>
<td>last</td>
<td>Recprobe2</td>
</tr>
<tr>
<td>Probe-no-vehicle</td>
<td>last</td>
<td>Recprobenov</td>
</tr>
<tr>
<td>cant-go</td>
<td></td>
<td>free agent, ln# = 0 Reccantgo</td>
</tr>
</tbody>
</table>

Name: Newfault
Admitted in: NE SA CA
Input: -
Requires: Messrec
If branch: Yes
Condition: Fault flag set? Speed? Stop ahead?

Effect: 1. If any fault flag set call Qtwofaults;
2. Endif; if maystop flag is set, ;
3. Else if speed << target, call Slowspeed;
4. Endif; endif; go to Actcalled;
A.5 BUSYCHECK
A.5.1 MERGE (Figure A.2)

Name: Busychek
Specification in: Sup (1)

Name: Callmerge
Admitted in: N NE SA
Input: -
Requires: Mersplit (10)
If branch: No
Effect: Send **Req_merge**; set merge(l)-flag = 1 in Busychek; reset flag in **noreqm1**; set busy; return;

Name: Contactm
Admitted in: N NE SA
Input: -
Requires: Sepmeasure and close spacing
If branch: Yes
Condition: Flag set?
Effect: 1. Send confirm-merge; set flag = 3;
2. If flag set set stime in norepm3 = time + (increment);
3. Else set stime in norepm3 = time;
4. **Endif**; return;

Name: Norepm 1
Admitted in: N NE SA
Input: -
Requires: Recnackreqm (no messages)
If branch: Yes
Condition: Flag? long delay?
Effect: 1. If flag not set, set flag; stime = time;
2. Else if (time - stime) excessive, reset busy; call Toolong;
3. Else no action; **endif**; **endif**;
4. return;

Name: Norepm2
Admitted in: N NE SA
Input: -
Requires: Recconmer
If branch: Yes
Condition: Flag? long delay?
Effect: 1. If flag not set, set flag; stime = time;
2. Else if (time - stime) excessive, reset busy; call Toolong;
3. Else no action; **endif**; **endif**;
4. return;
Figure A.2. Merge.
Name: Norepm3
Admitted in: N NE SA
Input: -
Requires: Buschek merge(l) Flag = 3
If branch: Yes
Condition: (Time - stime excessive)?
Effect: 1. If (time - stime) not excessive, no action;
2. Else set split(2); reset merge(l);
3. endif; return;

Name: Recackreqm
Admitted in: N NE SA
Input: Message flags
Requires: Buschek - merge(l) flag = 1
If branch: Yes
Condition: Ack_request_merge rec’d?
Effect: 1. If ack_request_merge received, increment target speed; Set
merge(1)-flag=2; reset message flag; reset flags in Noreqm1, Contactm;
return;
2. Else call Recnackreqm; return; endif;

Name: Recackreqmerbeh
Admitted in: N NE SA
Input: Message Ack_request_merge
Requires: Messrec pos = last
If branch: No
Effect: Set merge(3): set busy; return;

Name: Reccon_mer
Admitted in: N NE SA
Input: -
Requires: Buschek - merge(2)
If branch: Yes
Condition: Confirm-merge received?
Effect: 1. If confirm-merge received, update ownsize and transmit new control data,
pltn#, etc; reset busy; reset message flag; reset flag in norepm2; reset
merge(2);
2. Else call norepm2;
3. Endif; return;
Recnackreqm
Admitted in: N NE SA
Input: Message flags
Requires: Recnackreqm

If branch: Yes

Effect: 1. If nack_request_merge received, reset busy; reset message flag; reset flag in Norepm 1; reset merge(1);
2. Else call Norepm1; endif;
3. return;

Recreqmer3set
Admitted in: N SA
Input: Message Request-merge-3
Requires: Messrec

If branch: No

Effect: Set merge(3); set busy; return;

Recreqmer
Admitted in: N NE SA
Input: Message request-merge
Requires: Messrec

If branch: Yes

Condition: Same lane? Busy?

Effect: 1. If in same lane, then
2. If not busy and no fault set, call Sendackreqmer;
3. Else call Sendnackreqmer;
4. Endif; endif; return;

Recunmerge
Admitted in: N NE SA
Input: -
Requires: Buschek, merge(l), flag = 2

If branch: Yes

Condition: Unmerge rec’d? Delmerge rec’d?

Effect: 1. If unmerge received, reset message flag; set split(2);
2. Else if delmerge received, reset message flag; set flag in contactm;
3. Else call sepmeasurem;
4. Endif; endif; return;
Name: S t i l l g o
Admitted in: N NE SA
Input: -
Requires: Sepmeasure
If branch: Yes
Effect: 1. If flag not set, set flag; set stime = time;
       2. Else if (time - stime) not excessive, no action
       3. Else call Toolong; endif; endif;
       4. return;

Name: S e p m e a s u r e
Admitted in: N NE SA
Input: -
Requires: Recunmerge
If branch: Yes
Condition: Separation < 1.5 *(inplatoon separation)?
Effect: 1. Reads separation from veh ahead on probe.
       2. If this less than (normal in-platoon separation)*1.5 call Stillgo;
       3. Else call Contactm;
       4. Endif; return;

Name: S endn a c k r e q m e r
Admitted in: N NE SA
Input: -
Requires: Recreqmer and busy
If branch: No
Condition: -
Effect: Send nack_request_merge; return;

Name: S endackreqmer
Admitted in: N NE SA
Input: -
Requires: Recreqmer and not busy
If branch: No
Condition: -
Effect: 1. Send ack_request_merge;
       2. Set busy; set merge(2); reset flag in norepm2;
       3. return;

Name: S tillgo
Admitted in: N NE SA
Input: -
Requires: Sepmeasure
If branch: Yes
Condition: Flag set?
Effect: 1. If flag not set, set flag; set stime = time;
       2. Else if (time - stime) not excessive, no action
       3. Else call Toolong; endif; endif;
       4. return;
Name: Toolong  
Admitted in: N NE SA CA  
Input: -  
Requires: Several precursors  
If branch: No  
Condition: -  
Effect: Send message too-long to system (includes state vector, and flags in Buschek); reset busy; reset merge and split flags in Buschek; return;

Note. System will advise controllers - this can be a system fault, but presents no immediate danger.
A.5.2 SPLIT (Figure A.3)

Name: Busychek

Specification in: Sup (1)

Name: Callsplit

Admitted in: N NE SA CA
Input: 
Requires: Mersplit (5) or Linkmess (5)
If branch: Yes
Effect: 1. Sets busy;
       2. If pos = 1, call invitesplit;
       3. Else call reqsplit;
       4. Endif; return;

Name: Invitesplit

Admitted in: N NE SA CA
Input: 
Requires: Callsplit, pos = 1
If branch: No
Condition: Pos = 1?
Effect: Send invite-split; Set split (1); reset flag in norepsl; update state vector;
return;

Name: Newlead

Admitted in: N NE SA CA
Input: 
Requires: Buschek, split (2) set, flag = 2.
If branch: No
Condition: 
Effect: Decrement target speed. set pos = 1; update state vector; Set flag = 3; Reset flags in spliton and noreps2; return;

Name: Noreps 1

Admitted in: N NE SA CA
Input: 
Requires: Recconsplit (no messages)
If branch: Yes
Condition: Flag? long delay?
Effect: 1. If flag not set, set flag; stime = time;
       2. Else if (time - stime) excessive, call Toolong;
       3. Else no action; endif; endif;
       4. return;
Figure A.3. Split and Split-change-lane.
Name: Noreps2  
Admitted in: N NE SA CA  
Input: -  
Requires: Recnackreqspl (no messages)  
If branch: Yes  
Condition: Flag? long delay?  
Effect:  
1. If flag not set, set flag; stime = time;  
2. Else if (time - stime) excessive, call Toolong;  
3. Else no action; endif; endif;  
4. return;  

Name: Noreps3  
Admitted in: N NE SA CA  
Input: -  
Requires: Recsplitover (no messages)  
If branch: Yes  
Condition: Flag? long delay?  
Effect:  
1. If flag not set, set flag; stime = time;  
2. Else if (time - stime) excessive, call Toolong;  
3. Else no action; endif; endif;  
4. return;  

Name: Recackreqsplit  
Admitted in: N NE SA CA  
Input: -  
Requires: Buschek, split(2) set, flag = 1.  
If branch: Yes  
Condition: Ack_req_split received?  
Effect:  
1. If ack_request_split received, reset message flag; set flag = 2; reset flag in noreps2;  
2. Else call Recnackreqsplit;  
3. Endif; return;  

Name: Recakrqsbeh  
Admitted in: N NE SA CA  
Input: Message Ack_request_split or invite-split to vehicle behind  
Requires: Messrec  
If branch: No  
Condition: -  
Effect: Reset message flag; set busy; set split(3); reset flag in noreps3; return;
Name: Reconsplit  
Admitted in: N NE SA CA
Input: -
Requires: Buschek, split(1) set
If branch: Yes  
Condition: confirm-split received?
Effect:  
1. If confirm-split received, reset message flag; reset busy; send split-over; reset flag in norepsl; reset split(l);
2. Else call noreps1 ;
3. Endif; return;

Name: Reqsplit  
Admitted in: N NE SA CA
Input: -
Requires: Callsplit, pos > 1
If branch: No  
Condition: -
Effect: Send request-split; set split(2); set flag = 1; reset flag in noreps2; return;

Name: Recinv_spl  
Admitted in: N NE SA CA
Input: Message invite-split
Requires: Messrec
If branch: No  
Condition: -
Effect: Reset message flag; set busy; set split(2) - flag = 2; return;

Name: Recnackreqs  
Admitted in: N NE SA CA
Input: -
Requires: Recackreqs, ack not received
If branch: Yes  
Condition: Nack_request_split received?
Effect:  
1. If nack_request_split received, reset busy; reset message flag; reset split(2); reset flag in noreps2;
2. Else call noreps2;
3. endif; return;

Name: Recreqsplit  
Admitted in: N NE SA CA
Input: Message request-split
Requires: Messrec and pos = 1;
If branch: Yes  
Condition: Same lane? Busy?
Effect:  
1. If same lane, then
2. If busy or fault set, call Sendnackreqs;
3. Else call Sendackreqs;
4. Endif; endif; return;
Name: Reqsplchl . Fig: 3, 8
Admitted in: N SA
Input: Message Req_spl_change_lane
Requires: Messrec
If branch: No
Effect: Set split(2); set split(2)-flag = 2; reset flag in noreps2; set busy; return;

Name: Reqsplchlbeh Fig: 3, 8
Admitted in: N SA
Input: Message request-split-change-lane rec’d for vehicle behind
Requires: Messrec
If branch: No
Effect: Set split(3); set busy; reset flag in noreps3; return;

Name: Recsplitover Fig: 3
Admitted in: N NE SA CA
Input: -
Requires: Buschek, split(3).
If branch: Yes
Effect: 1. If split-over received, reset message flag; reset busy; reset flag on noreps3; reset split(3);
2. Else call noreps3;
3. endif; return;

Name: Sendackreqs Fig: 3, 8
Admitted in: N NE SA CA
Input: -
Requires: Reqsplits
If branch: No
Effect: Send ack_request_split; set busy; set split(1); reset flag in noreps1; update state vector; return

Name: epmeasures Fig: 3
Admitted in: N NE SA CA
Input:
Requires: Buschek, split(2)-flag = 3.
If branch: No
Effect: Read from probe separation from vehicle ahead; pass this in calling Splitdone; return;
Name: Sendnackreqs  
Admitted in: N NE SA CA  
Input: -  
Requires: Recreqsplit and busy set  
If branch: No  
Effect: Send \texttt{nack\_request\_split}; return  

Name: Spliton  
Admitted in: N NE SA CA  
Input: -  
Requires: Splitdone (no messages)  
If branch: Yes  
Condition: Flag? long delay?  
Effect: 1. If flag not set, set flag; stime = time;  
2. Else if (time - stime) excessive, call \texttt{Toolong};  
3. Else no action; \texttt{endif}; \texttt{endif};  
4. return;

Name: Splitdone  
Admitted in: N NE SA CA  
Input: Distance from veh ahead (= d)  
Requires: Sepmeasures,  
If branch: Yes  
Condition: d < platoon spacing?  
Effect: 1. If d < platoon spacing, call \texttt{spliton};  
2. Else send confirm-split; reset busy; reset flag in \texttt{Spliton}; reset split(2);  
3. \texttt{endif}; return;

Name: Toolong  
Specification in: Merge (2).
AS.3 CHANGELANE (Figure A.4)

Name: Busychek  
Specification in: Sup (1)  
Fig: 1,2,3,4,5,6

Name: Callchangelane  
Admitted in: N NE SA CA  
Input: -  
Requires: Linkmess (10)  
If branch: No  
Effect: Set busy; set chng(1); set flag = 1; call Reqchngln; return;

Name: Chltomer2  
Admitted in: N SA  
Input: -  
Requires: Sendconchl2  
If branch: No  
Effect: Reset chng(1); set merge(2); return;

Name: Compposns  
Admitted in: N SA  
Input: -  
Requires: Buschek, chng(2) and flag = 4  
If branch: Yes  
Condition: Results of calculation  
Effect:  
1. From pos, speed of entering car, own speed, ownsize, pos and data sent, determine change-lane strategy;  
2. If veh is to join behind, call Sendreqdecel; flag = 5;  
3. Else if in centre, call Sendreqsplchll; flag = 5;  
4. Else ahead, call Dropback; flag = 6;  
5. Endif; endif; return;

Name: Dropback  
Admitted in: N SA  
Input: -  
Requires: Compposns  
If branch: No  
Condition:  
Effect:  
1. Determines position and velocity of both vehicles so that enterer will arrive at gate at head of platoon;  
2. Records targets, stores them; flag = 6; return;
Figure A.4. Change Lane (initiator).

(t-p = turning point)
Figure A.4a. Change Lane (recipient).
(t-p = turning point)
Name: Forsplitl
Admitted in: N NE SA CA
Input: -
Requires: Recconsplit
If branch: Yes
Condition: Forced split called? sect in NE?
Effect: 1. If forced split called (ie forspl(1) in buschek set), send forced-split-called; reset chng(2);
2. Else if section is now in NE or CA, send-forced-split-called; reset chng(2); reset busy;
3. Endif; call Recforsplcall3; return;

Name: Forsplit2
Admitted in: N NE SA CA
Input: -
Requires: Sendcondecel
If branch: Yes
Condition: Forced split called? sect in NE?
Effect: 1. If forced split called (ie forspl(1) in buschek set), send forced-split-called; reset chng(2);
2. Else if section is now in NE or CA, send-forced-split-called; reset chng(2); reset busy;
3. Endif; call Recforsplcall4; return;

Name: Norepch 1
Admitted in: N NE SA CA
Input: -
Requires: Recforsplcall2
If branch: Yes
Condition: Gate passed? Counter > 1?
Effect: 1. If gate signal received, then if signal indicates section of receiver is in NE, reset busy;
2. Else counter += 1; if counter > 1, then reset busy; advise system - no confirmation from platoon . . . ;
3. Endif; endif; endif; return;
Name: Norepch2
Admitted in: N SA
Input: -
Requires: Recforsplcall3
If branch: Yes

Condition: Gate present? Counter > 1?
Flag?

Effect: 1. If gate present, then
2. If flag reset, set flag; counter += 1;
3. Endif; Else if counter > 1, reset busy; reset chng(2); advise system;
4. Else if flag set, reset flag;
5. Endif; endif; endif; return;
Name: Norepch3  
Admitted in: N SA  
Input: -  
Requires: Reconchncl2  
If branch: Yes  
Condition: Gate present? Counter >0?  
Flag?

Effect: 1. If gate present, then  
2. If flag reset, set flag; counter += 1;  
3. Endif; Else if counter > 1, reset busy; reset chng(2); advise system;  
4. Else if flag set, reset flag;  
5. Endif; endif; endif; return;

Name: Norepch4  
Admitted in: N SA  
Input: -  
Requires: Reconchncl1  
If branch: Yes  
Condition: Gate present?

Effect: 1. If gate present, ;  
2. Else reset busy; reset chng(2); advise system;  
3. Endif; return;

Name: Norepch5  
Admitted in: N SA  
Input: -  
Requires: Reconchncl3  
If branch: Yes  
Condition: Gate present? Counter >2?  
Flag?

Effect: 1. If gate present, then  
2. If flag reset, set flag; counter += 1;  
3. Endif; Else if counter > 2, reset busy; reset chng(2); advise system;  
4. Else if flag set, reset flag;  
5. Endif; endif; endif; return;
Name: Pass1
Admitted in: N NE SA CA
Input: -
Requires: Buschek, chng(1) and flag = 8
If branch: Yes

Effect: 1. If gate detected, then
2. If flag 1 set, then
3. If t-p detected, turn to pass gate; update state vector; set flag 2;
4. Endif; else set flag 1;
5. Endif; else if flag 2 set; flag (buschek) = 11;
6. Else if flag 1 set, call Syscall2; reset busy;
7. Endif; endif; endif; return;

Name: Pass2
Admitted in: N NE SA CA
Input: -
Requires: Buschek, chng(1) and flag = 6
If branch: Yes

Effect: 1. If gate detected, then
2. If flag 1 set, then
3. If t-p detected, turn to pass gate; update state vector; set flag 2;
4. Endif; else set flag 1;
5. Endif; else if flag 2 set; flag (buschek) = 9;
6. Else if flag 1 set, call Syscall1; reset busy;
7. Endif; endif; endif; return;

Name: Pass3
Admitted in: N NE SA CA
Input: -
Requires: Buschek, chng(1) and flag = 7
If branch: Yes

Effect: 1. If gate detected, then
2. If flag 1 set, then
3. If t-p detected, turn to pass gate; update state vector; set flag 2;
4. Endif; else set flag 1;
5. Endif; else if flag 2 set; flag (buschek) = 10;
6. Else if flag 1 set, call Syscall1; reset busy;
7. Endif; endif; endif; return;
Name: Recackreqchl
Admitted in: N NE SA CA
Input: -
Requires: Recnackreqchl
If branch: Yes
Effect: 1. Send \texttt{thanx\_but\_no} to all irrelevant replies;
2. If relevant reply received from adjacent lane, store stated speed; set
flag = 4; reset counter in norepchl;
3. Else if reply only from next lane, store stated speed;
4. Endif; endif; reset all message flags; return;

Name: Reccondecel
Admitted in: N NE SA CA
Input: -
Requires: Recconsplchl
If branch: Yes
Effect: 1. If confirm-decel received,
2. Reset message flag; if in position,
3. Flag = 7; reset both flags in Pass3;
4. Else set fault #11;
5. Endif; else call Recforsplcalll ;
6. Endif; return;

Name: Recconchll
Admitted in: N SA
Input: -
Requires: Buschek, chng(2) and flag = 7
If branch: Yes
Effect: 1. If confirm-change-lane received, reset chng(2);
2. Endif; else call Recunconchll;
3. Endif; return;

Name: Recconchll2
Admitted in: N SA
Input: -
Requires: Buschek, chng(2) and flag = 7
If branch: Yes
Effect: 1. If confirm-change-lane received, send mer3set; reset chng(2); set merge(2);
2. Else call Recunconchll2;
3. Endif; return;
Name: Recconch13  
Admitted in: N SA  
Input: -  
Requires: Buschek and chng(3)  
If branch: Yes  
Condition: Confirm-change-lane received?  
Effect: 1. If confirm-change-lane received, reset busy; reset chng(3); reset message marker;  
2. Else call Rectbn2;  
3. Endif; return;  

Name: Reccondropt  
Admitted in: N NE SA CA  
Input: -  
Requires: Buschk, Chng(2), flag = 5  
If branch: Yes  
Condition: Confirm-dropt received?  
Effect: 1. If confirm-dropt received, then if in position,  
   2. set flag = 7; reset counter in Norepch3; reset message flag;  
   3. Else send forsplit-called;  
   4. Endif; else adjust speed to meet target; call Recconsplit;  
   5. Endif; return;  

Name: Recconsplit  
Admitted in: N NE SA CA  
Input: -  
Requires: Reccondropt  
If branch: Yes  
Condition: Confirm-split received?  
Effect: 1. If confirm-split received,  
   2. If near gate, reset message flag; send split-over;  
   3. If in position, send confirm-split-change-lane; reset counter in Norepch3; flag = 7;  
   4. Else send for-split-called;  
   5. Endif; else continue to achieve position; call Forsplitl;  
   6. Endif; else call Forsplitl;  
   7. Endif; return;
Name: Recconsplitchl Fig: 4
Admitted in: N NE SA CA
Input: -
Requires: Recreqdecel
If branch: Yes Condition: Confirm-split-chng-In received?
Effect: 1. If confirm_split_change_lane received,
2. Reset message flag; if in position,
3. Flag = 6; reset both flags in pass2;
4. Else set fault #11;
5. Endif; else call Reccondecel;
6. Endif; return;

Name: Rectbn 1 Fig: 4
Admitted in: N SA
Input: -
Requires: Repwait2
If branch: Yes Condition: Thanx-but-no received:
Effect: 1. If thanx-but-no received, reset busy; reset message marker; reset chng(2);
2. Endif; return;

Name: Rectbn2 Fig: 4
Admitted in: N SA
Input: -
Requires: Recconchl3
If branch: Yes Condition: Thanx-but-no received:
Effect: 1. If thanx-but-no received, reset busy; reset message marker; reset chng(3);
2. Else call Recunconchl3;
3. Endif; return;

Name: Repwait 1 Fig: 4
Admitted in: N NE SA CA
Input: -
Requires: Buschek, chnge(1), flag = 1, 2, or 3.
If branch: No Condition: -
Effect: Call Recnackch; flag + = 1; return;

Name: Repwait2 Fig: 4
Admitted in: N SA
Input: -
Requires: Buschek, chnge(2), flag = 1, 2, or 3.
If branch: No Condition: -
Effect: Call Rectbnl; flag + = 1; return;
Name: Recforsplcall1
Admitted in: N NE SA CA
Input: -
Requires: Voidch, Sendcondropt
If branch: Yes
Condition: Forced-split-called received?
Effect: 1. If forced_split_called received, reset busy; reset message flag;
2. endif; return;

Name: Recforsplcall2
Admitted in: N NE SA CA
Input: -
Requires: Reccondecel
If branch: Yes
Condition: Forced_split_called received?
Effect: 1. If forced_split_called received, reset message flag; reset busy;
2. Else call Norepchl;
3. endif; return

Name: Recforsplcall3
Admitted in: N NE SA CA
Input: -
Requires: Forsplit1
If branch: Yes
Condition: Forced-split-called received?
Effect: 1. If forced_split_called received, reset busy; reset message flag;
2. Endif; call norepch2; return;

Name: Recforsplcall4
Admitted in: N NE SA CA
Input: -
Requires: Forsplit2
If branch: Yes
Condition: Forced_split_called received?
Effect: 1. If forced-split-called received, reset busy; reset message flag;
2. endif; return;
Name: Recnackreqchln
Admitted in: N NE SA CA
Input: -
Requires: Repwait
If branch: Yes

Effect: 1. If flag = 3, then
2. If veh in adjacent lane has sent \texttt{nack\_request\_chng\_ln}, send thanx-but-no to all those who have replied, except the \texttt{nack\_er}; reset busy;
3. Else if no replies from adjacent lane, and veh in lane two away has sent \texttt{nack\_request\_change\_lane} send thanx-but-no to all others; reset busy; reset all message flags;
4. Else call Recreqackchln;
5. Endif; endif; endif; return;

Name: Reqchngln
Admitted in: N NE SA CA
Input: -
Requires: Callchangelane
If branch: No

Effect: Send \texttt{req\_chng\_lane}, including \texttt{loc} and maxspeed; return;
Name: Recreqchnln .  
Admitted in: N NE SA CA S C  
Input: Loc and maxspeed of sender.  
Requires: Messrec  
If branch: Yes  
Condition: Within 2 lanes on side of change? Busy? in N or SA?  
Effect: 1. If in adjacent or next lane on side of change, and within platoon spacing of loc, then  
2. If “busy” not set, and mode is N or SA, and no fault set, call Sendackreqchln with data received;  
3. Else call Sendnackreqchln;  
4. Endif; endif; return;

Name: Recreqdecel;  
Admitted in: N NE SA CA  
Input: -  
Requires: Buschek, chng(1), flag = 5.  
If branch: Yes  
Condition: Request-decel received?  
Effect: 1. If request-decel received, flag = 11; store target time of arrival at gate; reset message flag;  
2. Else call Recconspl; set speed to stated speed of requester;  
3. Endif; return;

Name: Recunconchl1  
Admitted in: N SA  
Input: -  
Requires: Recconchl2  
If branch: Yes  
Condition: Unconfirm_change_ln received?  
Effect: 1. If unconfirm change-lane received, reset chng(2); reset busy; reset message flag;  
2. Else call Norepch4;  
3. Endif; return;

Name: Recunconchl2  
Admitted in: N SA  
Input: -  
Requires: Recconchl2  
If branch: Yes  
Condition: Unconfirm_change_ln received?  
Effect: 1. If unconfirm change-lane received, reset chng(2); reset busy; reset message flag;  
2. Else call Norepch3;  
3. Endif; return;
Name: Recunconchl3
Admitted in: N SA
Input: -
Requires: Tectbn2
If branch: Yes
    Condition: Unconfirm_chng_in received?
Effect: 1. If unconfirm_change_lane received, reset busy; reset message flag; reset chng(3);
        2. Else call norepch5;
        3. Endif; return;

Name: Sendackreqchln
Admitted in: N SA
Input: loc and maxspeed of requester
Requires: RecreqchngLn, not busy and right mode
If branch: Yes
    Condition: Lane adjacent? next to this?
Effect: 1. Set new speed - minimum of own and requester’s maxspeed;
        2. Send Ack_request_change_lane with lane and new speed; set busy; store data sent and received;
        3. If in adjacent lane, set chng(2); set flag = 1;
        4. Else set chng(3); reset counter and flag in Norepch5;
        5. Endif; return;

Name: Sendconchln 1
Admitted in: N SA
Input: -
Requires: Buschek, chng(1) and flag = 9
If branch: Yes
    Condition: Gap too big?
Effect: 1. Send confirm-change-lane to all participating agents;
        2. If gap ahead too big; call Syscall3;
        3. Endif; call Chltomer1; return;

Name: Sendconchln2
Admitted in: N NE SA CA
Input: -
Requires: Buschek, chng(1) and flag = 10
If branch: No
    Condition: -
Effect: Send confirm-change-lane to all participating agents; call chltomer1; return;
Name: Sendcondecel
Admitted in: N NE SA CA
Input: -
Requires: Buschek, chng(2), flag = 6
If branch: Yes
Effect: 1. Adjust speed to reach gate at target time;
   2. If gate near, then if in position,
   3. Send confirm-decel; flag = 8;
   4. Else send forsplit-called;
   5. Endif; else call forsplit2;
   6. Endif; return;

Name: Sendcondropt
Admitted in: N NE SA CA
Input: -
Requires: Buschek, chng(1) and flag = 12
If branch: Yes
Effect: 1. Adapt speed to reach gate at target time;
   2. If gate reached,
   3. If in position, flag = 6; send confirm-dropt; reset flags in pass2;
   4. Else send forsplit-called;
   5. Endif; else call Recforsplcalll;
   6. Endif; return;

Name: Sendconvoid
Admitted in: N SA
Input: Reading from probe
Requires: Buschek, chng(1) and flag = 11
If branch: Yes
Effect: 1. Send confirm-change-lane to any participating agent;
   2. If vehicles ahead in platoon spacing detected, advise system;
   3. Endif; reset busy; reset chng(1); return;

Name: Sendnackreqchln
Admitted in: N NE SA CA S C Q
Input: -
Requires: Recreqchln and (busy or wrong mode)
If branch: Yes
Effect: Send nack_request_change_lane; return;

Name: Sendreqdecel
Admitted in: N SA
Input: -
Requires: Compposns
If branch: No

Effect: 1. Determines position and velocity of both vehicles so that enterer will arrive at gate at back of platoon;
2. Records targets, stores them; sends request-decel with this data;
3. Resets counter, flag in norepch2; flag = 5;
4. return;

Name: Sendreqsplchl
Admitted in: N SA
Input: -
Requires: Compposns
If branch: No

Effect: 1. Determines position and velocity of both vehicles so that enterer will arrive at gate behind vehicle (N-l) of platoon;
2. Records targets, stores them; sends request-split-change-lane to vehicle N;
3. Resets counter, flag in norepch2; flag = 5;
4. return;

Name: Syscall1
Admitted in: N NE SA CA
Input: -
Requires: Pass2, pass3, and no t-p
If branch: No

Effect: Send unconfirm_change_lane; Send to system noTp_1; return;

Name: Syscall2
Admitted in: N NE SA CA
Input: -
Requires: Pass1, and no t-p
If branch: No

Effect: Send unconfirm-change-lane; send to system noTp_2; set fault #6; return;
Name: Syscall3
Admitted in: N SA
Input: -
Requires: Sendconchll or Recconchll and big gap
If branch: No
Effect: Send change-messed-up (with ID’s of participants); return;

Name: Voidchange
Admitted in: N NE SA CA
Input: -
Requires: Buschek, chng(1), flag = 4
If branch: Yes
Effect: 1. If gate near (as determined by loc and its broadcast of its loc), send change-to-void; flag = 8; reset both flags in Passl;
2. Else call Recforspicall1;
3. Endif; return;
AS.4 FORCED-SPLIT (Figure A.5)

**Name:** Buslead 1
**Admitted in:** N NE SA CA
**Input:** -
**Requires:** Fsl set, \((\text{pos} = 1)\) and busy
**If branch:** Yes

**Condition:** multiple

**Effect:** 1. Action according to flag set in buschek:
- Merge(1): set forsplit(4); store data; set active flag; break;
- Merge(2): reset merge(2); set forsplit(1); set time in Norepforsl; break;
- Split(1): reset split(1); set forsplit(1); set time in Norepforsl; break;
- Split(2): set forsplit(5); store data; set active flag; break;
- Chng(2): set forsplit(1); set time in Norepforsl; break;
- Chng(3): store data; set active flag; set forsplit(1); set time in Norepforsl; break;
- Emerch(x), \(x = 2,3,4,5\) set forsplit(1); set time in Norepforsl; advise system; break;
- Forspl(x), \(x = 1,6\): Call stop mode; advise system;

2. endifs; return;

**Name:** Buslead2
**Admitted in:** N NE SA CA
**Input:** -
**Requires:** Leaqforsplitl and busy
**If branch:** Yes

**Condition:** multiple

**Effect:** 1. Action according to flag set in buschek:
- Merge(1): set forsplit(4); store data; set active flag; break;
- Merge(2): reset merge(2); set forsplit(1); set time in Norepforsl; break;
- Split(1): reset split(1); set forsplit(1); set time in Norepforsl; break;
- Split(2): set forsplit(5); store data; set active flag; break;
- Chng(2): set forsplit(1); set time in Norepforsl; break;
- Chng(3): store data; set active flag; set forsplit(1); set time in Norepforsl; break;
- Emerch(x), \(x = 2,3,4,5\) set forsplit(1); set time in Norepforsl; advise system; break;
- Forspl(x), \(x = 1,6\): Call stop mode; advise system;

2. endifs; return;
Figure AS. Forced Split (initiation).
Figure A.5a. Forced Split (implementation).
Name: Buslead3  
Admitted in: N NE SA CA  
Input: -  
Requires: Leadforsplit2 and busy  
If branch: Yes  
Condition: multiple  
Effect: 1. Action according to flag set in buschek;
   Merge(l): set forsplit(4); store data; set active flag; break;
   Merge(2): reset merge(2); set forsplit(1); set time in Norepforsl; break;
   Split(l): reset split(l); set forsplit(1); set time in Norepforsl; break;
   Split(2): set forsplit(5); store data; set active flag; break;
   Chng(2): set forsplit(1); set time in Norepforsl; break;
   Chng(3): store data; set active flag; set forsplit(1); set time in Norepforsl; break;
   Emerch(x), x = 2,3,4,5 set forsplit(1); set time in Norepforsl; advise system; break;
   Forspl(x), x = 1,6: Call stop mode; advise system;
2. endifs; return;

Name: Buslead4  
Admitted in: N NE SA CA  
Input: -  
Requires: Forspll and busy  
If branch: Yes  
Condition: multiple  
Effect: 1. Action according to flag set in buschek;
   Merge(l): set forsplit(4); store data; set active flag; break;
   Merge(2): reset merge(2); set forsplit(1); set time in Norepforsl; break;
   Split(l): reset split(l); set forsplit(1); set time in Norepforsl; break;
   Split(2): set forsplit(5); store data; set active flag; break;
   Chng(2): set forsplit(1); set time in Norepforsl; break;
   Chng(3): store data; set active flag; set forsplit(1); set time in Norepforsl; break;
   Emerch(x), x = 2,3,4,5 set forsplit(1); set time in Norepforsl; advise system; break;
   Forspl(x), x = 1,6: Call stop mode; advise system;
2. endifs; return;

Name: Busychek  
Specification in: Sup (1)

Fig: 5, 8
Fig: 5, 10
Fig: 1,2,3,4,5,6

A-45
Name: Forspla
Admitted in: N NE SA CA
Input: -
Requires: Actcalled
If branch: No
Effect: Set busy; set forspl(2); set flag = 1; call Sendaheadb; call Sendbehindb; return;

Name: Forspll
Admitted in: N NE SA CA
Input: -
Requires: Actcalled
If branch: Yes
Condition: Busy?
Effect: 1. If busy, call Buslead4;
2. Else set busy; set forspl(1); set time in Norepforsl;
3. Endif; call Sendbehindl; return;

Name: Fs 1 set
Admitted in: N NE SA CA
Input: -
Requires: Sendbehinda and pos = 1
If branch: Yes
Condition: Busy?
Effect: 1. If busy, call Buslead1;
2. Else set busy; set forspl(1); set time in Norepforsl;
3. Endif; return;

Name: Fs2set
Admitted in: N NE SA CA
Input: -
Requires: Sendaheadbl and not busy, sendbehindbl
If branch: No
Condition: -
Effect: Set busy; set forspl(2); set flag = 1; return;

Name: Fs3set
Admitted in: N NE SA CA
Input: -
Requires: Sendaheada or (Sendbehinda and pos > 1)
If branch: No
Condition: -
Effect: Set busy; set forspl(3); set time in norepfors2; return;
Name: Forsplitdone
Admitted in: N NE SA CA
Input: *
Requires: Sepmeasfs1 or Waitsep
If branch: Yes
Condition: Free agent?
Effect: 1. Send confirm-for-split;
   2. If free agent, set flag = 4;
   3. Else set flag = 5; set stime in Waitfast; reset flags in Splitagain and mergeagain;
   4. Endif; return;

Name: Histfor
Admitted in: N NE SA CA
Input: *
Requires: Recfsplitover2
If branch: Yes
Condition: More messages stored?
Effect: 1. If more messages stored, send forward oldest; delete it from store;
   2. Else reset busy; reset flag in recfsplitover2;
   3. endif; return;

Name: Lastbusy
Admitted in: N NE SA CA
Input: forspl_b(pos-x)_r (x > 1) received
Requires: Messrec and pos = last
If branch: Yes
Condition: Busy? and multiple.
Effect: 1. If busy, act on flag in buschek:
   Merge(3): send Unmerge; store data; set forspl(6); set time in Recfsplitover2;
   reset flag in Recfsplitover2; break;
   Split(3): store data; set forspl(6); set time in Recfsplitover2; reset flag in
   Recfsplitover2; break;
   Forspl(6): Inform system; call stop mode;
   2. Endifs; endif; return;

Name: Leadforsplit 1
Admitted in: N NE SA CA
Input: Message Forsplit-behind_ 1_f received by leader (pos = 1)
Requires: Messrec
If branch: Yes
Condition: Busy?
Effect: 1. Setforspl( 1); set time in Norepfors1;
   2. If busy call Buslead2;
   3. Else set busy;
   4. Endif; return;
Name: Leadforsplit2
Admitted in: N NE SA CA
Input: Forsplit\_behind\_(N>1)\_f received by leader
Requires: Messrec
If branch: Yes
  Condition: Busy?
Effect: 1. Set forspl(1); set time in Norepforsl;
       2. If busy call Buslead3;
       3. Else set busy;
       4. Endif; return;

Name: Mergeagain
Admitted in: N NE SA CA
Input: -
Requires: Splitagain
If branch: Yes
  Condition: Flag set? merge-hist received?
Effect: 1. If flag set, then if more stored history, read it;
       2. If item received is confirm-merge,
       3. Reset busy; reset forspl(2); reset flag; set target speed back to normal;
       4. Endif; else set merge(2); reset flag; set target speed back to normal;
       5. Endif; else if item received is merge-hist, set flag;
       6. Endif; return;

Name: Newlead
Admitted in: N NE SA CA
Input: -
Requires: Buschek, forspl(2), flag = 1
If branch: Yes
  Condition: Fault#1 set?
Effect: 1. Update state vector; reduce target speed;
       2. If fault #1 set, flag = 3; time to waitsep;
       3. Else flag = 2;
       4. Endif; return;

Name: Norepfors1
Admitted in: N NE SA CA
Input: -
Requires: Recconspll
If branch: Yes
  Condition: (time - stime) excessive?
Effect: 1. If (time - stime) excessive, reset busy, reset forspl(1); send for-split-over;
       2. Endif; return;

Name: Norepfors2
Admitted in: N NE SA CA
Input: -
Requires: Recsplitoverl
If branch: Yes \hspace{2cm} \textbf{Condition:} (time - stime) excessive?

\textbf{Effect:} 1. If (time - stime) excessive, reset busy, reset forspl(3);
                2. \textbf{Endif; return;}

\textbf{Name:} Norepfors3 \hspace{1cm} \textbf{Fig:} 5
\textbf{Admitted in:} N NE SA CA
\textbf{Input:} -
\textbf{Requires:} Recsplitover2

\textbf{If branch:} Yes \hspace{2cm} \textbf{Condition:} (time - stime) excessive?
\textbf{Effect:} 1. If (time - stime) excessive, reset forspl(4); set merge(1); set flag = 2; send forsp-over; set flag in Stillgo;
                2. Else call sepmeasfs2;
                3. \textbf{Endif; return;}

\textbf{Name:} Norepfors4 \hspace{1cm} \textbf{Fig:} 5
\textbf{Admitted in:} N NE SA CA
\textbf{Input:} -
\textbf{Requires:} Recsplitover3

\textbf{If branch:} Yes \hspace{2cm} \textbf{Condition:} (time - stime) excessive?
\textbf{Effect:} 1. If (time - stime) excessive, reset forspl(4); set merge(1); set flag = 3; send confirm-merge; send forsp-over; set time in Norepm3; set target speed back to normal;
                2. \textbf{Endif; return;}

\textbf{Name:} Norepfors3 \hspace{1cm} \textbf{Fig:} 5
\textbf{Admitted in:} N NE SA CA
\textbf{Input:} -
\textbf{Requires:} Recsplitover2

\textbf{If branch:} Yes \hspace{2cm} \textbf{Condition:} (time - stime) excessive?
\textbf{Effect:} 1. If (time - stime) excessive, reset forspl(5); set split(2); set flag = 3; send fsplitover; set flag in Spliton;
                2. Else call sepmeasfs3;
                3. \textbf{Endif; return;}

A-49
Name: Norepfors6
Admitted in: N NE SA CA
Input: -
Requires: Recsplitover5
If branch: Yes
Effect: 1. If (time - stime) excessive, reset forspl(5); reset busy; send confirm-split; send forsp_over; set target speed back to normal;
      2. Endif; return;

Name: Qfalt 1
Admitted in: N NE SA CA
Input: -
Requires: Buschek and forspl(1) set
If branch: Yes
Condition: Mode SA or CA?
Effect: 1. If mode is SA or CA, send Crashstopbehind;
      2. Else call RecconfSpl1;
      3. Endif; return;

Name: Qmess
Admitted in: N NE SA CA
Input: -
Requires: Buschek and forspl(6)
If branch: Yes
Condition: Message received?
Effect: 1. If message received, store it and add it as data to call to Recfsplitover2;
      2. Else add Null mark to call;
      3. Endif; call Recfsplitover2; return;

Name: Qslowmode
Admitted in: N NE SA CA
Input: -
Requirements: Buschek and forspl(2), flag = 3 (therefore fault #1)
If branch: Yes
Condition: Mode SA or CA?
Effect: 1. If mode = CA or SA, send Crashstopbehind;
      2. Else call Waitsep;
      3. Endif; return;
Name: Recaheadb
Admitted in: N NE SA CA
Input: Message Forsplit_behind_(pos-1)-r received
Requires: Messrec
If branch: Yes
Effect: 1. Set forspl(2); set flag = 1;
2. If busy, call Sendunmerge2;
3. Else set busy;
4. Endif; reset message flag; return;

Name: Recbehinda
Admitted in: N NE SA CA
Input: Veh (pos > 1) receives fspl_b_{pos}_f
Requires: Messrec
If branch: No
Effect: Set busy; set forspl(3); set time in norepfors2; reset message flag; return;

Name: Recconfsll
Admitted in: N NE SA CA
Input: -
Requires: Qfaltl
If branch: Yes
Effect: Condition: Confirm-forsplit received?
1. If confirm-forsplit received, reset busy; reset message flag; reset forspl(1);
   send for-split-over;
2. Else call norepforsl;
3. Endif; return;

Name: Recconfspl2
Admitted in: N NE SA CA
Input: -
Requires: Buschek and forspl(4)
If branch: Yes
Effect: Condition: Confirm-forced-split received?
1. If confirm-forced-split received, reset forspl(4); set merge(l); set flag = 2;
   send fsplitover; reset flag in Stillgo; reset message flag;
2. Else call Norepfors3;
3. Endif; return;
Name: Recconfspl3
Admitted in: N NE SA CA
Input: -
Requires: Buschek and forspl(4), flags = 2
If branch: Yes
Effect: 1. If confirm-for-split received, reset forspl(4); set merge(1); set flag = 3; send confirm-merge; send forsp-over; set time in Norepm3; set target speed back to normal;
   2. Else call Norepfors4;
   3. Endif; return;

Name: Recconfspl4
Admitted in: N NE SA CA
Input: -
Requires: Buschek and forspl(5)
If branch: Yes
Effect: 1. If confirm-forced-split received, reset forspl(5); set flag = 3; send fsplitover; reset flag in Spliton; reset message flag;
   2. Else call Norepfors4;
   3. Endif; return;

Name: Recconfspl5
Admitted in: N NE SA CA
Input: -
Requires: Buschek and forspl(5), flag = 2
If branch: Yes
Effect: 1. If confirm-for-split received, reset forspl(5); reset busy; send confirm-split; send forsp-over; set target speed back to normal;
   2. Else call Norepfors6;
   3. Endif; return;

Name: Recfsplitoverl
Admitted in: N NE SA CA
Input: -
Requires: Buschek, forspl(3)
If branch: Yes
Effect: 1. If for-splitover received, reset busy, reset forspl(3); reset message flag;
   2. Else call norepfors2;
   3. Endif; return;
Name: Recfsplitover2, Admitted in: N NE SA CA
Input: Message received by Qmess or null
Requires: Qmess
If branch: Yes  Condition: For-split-over received?
Time elapsed? Flag set?
Effect: 1. If flag set, call Histfor;
2. Else if for-split-over received, set flag;
3. Else if (time - stime) excessive, set flag;
4. Endif; endif; return;

Name: Sendaheada Admitted in: N NE SA CA
Input: - Requires: Fault4or5 or Recackcont
If branch: Yes  Condition: Pos = 1?
Effect: 1. Send backward message forsplit_behind_(pos)_r;
2. If pos = 1, call Fslset;
3. Else call Fs3set; send forward message forsplit_behind_ (pos)_f;
4. Endif; return;

Name: Sendaheadb Admitted in: N NE SA CA
Input: - Requires: Forspla
If branch: No  Condition: -
Effect: Send forward message forsplit_behind_(pos-1)_f; return;

Name: Sendbehindb Admitted in: N NE SA CA
Input: - Requires: Forspla
If branch: No  Condition: -
Effect: Send back message forsplit-behind-@os-1)-r; return;
Name: Sendbehindbl
Admitted in: N NE SA CA
Input: -
Requires: Nocontmess (11)
If branch: Yes
Fig: 5, 11
Condition: Pos = last?
Effect: 1. Send forward message $\text{forsplit\_behind\_}\left(\text{pos-1}\right)\_f$;
2. If pos = last, then if busy call Sendunmerge1;
3. Else call Fs2set;
4. Endif; call fs2set; send back message $\text{forsplit\_behind\_}\left(\text{pos-1}\right)\_b$;
5. Endif; return;

Name: Sendbehind
Admitted in: N NE SA CA
Input: -
Requires: Forspill (and so pos = 1)
If branch: No
Condition: -
Effect: Send fspl\_b\_1 \_r back; return;

Name: Sepmeasfs1
Admitted in: N NE SA CA
Input: Reading, d, of forward probe
Requires: Buschek and forfpl(2), flag = 2
If branch: Yes
Condition: d = platoon spacing?
Effect: 1. If d > = platoon spacing call Forsplitdone;
2. Endif; return;

Name: Sepmeasfs2
Admitted in: N NE SA
Input: Distance d from preceding vehicle - probe reading
Requires: Norepfors3
If branch: Yes
Condition: d <1.5*(in-platoon spacing)?
Effect: 1. If d <1.5*(in-platoon spacing); set flag = 2; set time in Norepfors4 equal to that in Norepfors3;
2. Endif; return;

Name: Sepmeasfs3
Admitted in: N NE SA
Input: Distance d from preceding vehicle - probe reading
Requires: Norepfors5
If branch: Yes
Condition: d > = platoon spacing?
Effect: 1. If d > = platoon spacing; set flag = 2; set time in Norepfors6 equal to that in Norepfors5;
2. Endif; return;
Name: Splitagainl
Admitted in: N NE SA CA
Input: -
Requires: Testactflag and activity flag set (therefore free agent)
If branch: Yes
Condition: Act flag set? Split-hist received?
Effect: 1. If more stored history, read item;
2. If flag set, then if item is split-hist, reset flag;
3. Endif; else if item is confirm-split; reset busy; reset forspl(2); set target speed back to normal;
4. Endif;endif; else if flag set, ;
5. Else set split(l); reset flag; reset forspl(2); set target speed back to normal;
6. Endif;endif; return;

Name: Splitagain
Admitted in: N NE SA CA
Input: -
Requires: Buschek, forspl(2) flag = 5
If branch: Yes
Condition: Flag set? Split-hist received?
Effect: 1. If more stored history, read item;
2. If flag set; then if item is confirm-split, reset busy; reset forspl(2); set target speed back to normal;
3. Endif; else if item is split-hist, set flag;
4. Endif; endif; else if flag set, set split(l); reset flag; reset forspl(2); set target speed back to normal;
5. Endif; endif; return;

Name: Sendunmerge1
Admitted in: N NE SA CA
Input: -
Requires: Sendaheadbl and busy (implies pos = last)
If branch: Yes
Condition: Multiple
Effect: 1. Set forspl(2); set flag = 1;
2. Action according to flag in buschek:
   Merge(3): reset merge(3); send unmerge; break;
   Split(3): store data; set active flag; break;
   Forspl(6): inform system; call stop mode;
3. Endifs; return;

Name: Sendunmerge2
Admitted in: N NE SA CA
Input: -
Requires: Recaheadb and busy (implies pos = last)

A-55
If branch: Yes

Condition: Multiple

Effect: 1. Action according to flag in buschek:
   - Merge(3): reset merge(3); send Unmerge; break;
   - Split(3): store data; set active flag; break;
   - Forspl(6): inform system; call stop mode;

   2. Endifs; return;

Name: Testactflag

Admitted in: N NE SA CA

Input: -

Requires: Buschek and forspl(2), flag = 4 (therefore free agent)

If branch: Yes

Condition: Active flag set?

Effect: 1. If active flag set, call splitagain1;
   2. Else reset busy; reset forspl(2); target speed back to normal;
   3. Endif; return;

Name: Waitfast

Admitted in: N NE SA CA

Input: -

Requires: Mergeagain

If branch: Yes

Condition: (time - stime) excessive?

Effect: (Note: Wait interval is for message to pass down platoon and back measured in millisec.
   1. If (time - stime excessive) reset busy; reset forspl(2); target speed back to normal;
   2. Endif; return;

Name: Waitsep

Admitted in: N NE SA CA

Input: -

Requires: Qslowmode

If branch: Yes

Condition: (time - stime) excessive?

Effect: 1. If (time - stime) excessive, call Forsplitdone;
   2. Endif; return
AS.5 EMER-CHANGE (Figure A.6)

Name: Busychek
Specification in: Fig: 1,2,3,4,5,6

Name: Callstop
Fig: 6
Admitted in: NE CA
Input: -
Requires: Passemch
If branch: Yes
Condition: Flag? gate near? SA speed?
Effect: 1. If flag set, then if gate near;
2. Else if maxspeed is SA speed, call Stop mode; reset any lanes put into SA;
3. Else call Sacall for each lane affected;
4. Endif; send unconfirm_emer_change; message Emerch_fail to system; reset busy;
5. Else if gate near set flag;
6. Endif; endif; return;

Name: Compposn
Fig: 6
Admitted in: NE CA
Input: -
Requires: Buschek, Emerch(1) and flag = 7
If branch: Yes
Condition: result of calculation
Effect: From known speed and positions of other vehicles predict time at a loc some way ahead at which all partners should be in position; send this target to partners via messages Emerch_target; reset flag0 in Qallrec; if position sent is before next gate set flag1 in Qallrec = 0; else set flag1 = 1; endif; reset flag in neset3; Flag = 8; return;

Name: Emerchcall
Fig: 6, 10
Admitted in: N NE SA CA
Input: -
Requires: Actcalled
If branch: Yes
Condition: Fault #6 or #7?
Effect: 1. Set busy; set Emerch(1); set flag = 1; call Nesetl;
2. If fault #6 or fault #7 is present, reroute reception and transmission channels so that side transmission becomes system transmission and side reception system reception, set flag in Sendreqemerch;
3. Else reset flag in Sendreqemerch;
4. Endif; return;
Name: Forsplitcall
Admitted in: N NE SA CA
Input: -
Requires: Inposn
If branch: Yes
Effect: 1. If for-split called (ie \texttt{forspl(1)} in buschek set), send for-split-called to changer; reset emerch(x);
   2. Else call recforsplcal3;
   3. \texttt{Endif}; return;

Name: Inposn
Admitted in: N NE SA CA
Input: -
Requires: Buschek, emerch(x) (2 \leq x < 5) and flag = 7
If branch: Yes
Effect: 1. Call \texttt{forsplitcall1};
   2. If \texttt{loc} = target position, then
      3. If gate near, send message (message depends on x, see below); flag = 8; reset flag in Norepem3;
      4. \texttt{Endif}; \texttt{endif}; return;

Value of x  Message
2          adj\_ahead\_ok
3          adj\_behind\_ok
4          next\_ahead\_ok
5          next\_behind\_ok

Name: Neset 1
Admitted in: N NE SA CA
Input: -
Requires: Emerchcall
If branch:? Yes
Effect: 1. If fault \#6 or \#7 set, then if mode = CA, ;
   2. Else mode = NE from previous gate to one two ahead; \texttt{maxspeed} unchanged;
   3. \texttt{Endif}; \texttt{endif}; call Sendreqemerch; return;
Figure A.6. Emergency Change (initiator).
Figure A. 6a. Emergency Change (participant).
Name: Neset2
Admitted in: NE CA
Input: -
Requires: Recforsplitc12
If branch: Yes  
Condition: Faults? Flag?
Effect: 1. If fault #6 or #7, then if flag set, then if gate near, ;
2. Else set NE (and if necessary SA in parallel lanes) for two gates ahead; restore previous mode for section behind gate just passed;
3. Endif; else if gate near, set flag;
4. Endif; endif; endif; call Qallrec; return;

Name: Neset3
Admitted in: N SA
Input: -
Requires: Buschk. emerch(1) and flag = 10
If branch: Yes  
Condition: Faults?
Effect: 1. If fault #6 or #7, set new lane to NE mode, no change in maxspeed; reset lane to left to original mode and original speed; reset any lane set to SA to previous value;
2. Endif; call Sendconemerch; return;

Name: Norepm 1
Admitted in: N NE SA CA
Input: -
Requires: Recposn
If branch: Yes  
Condition: Flag? time excessive;
Effect: 1. If flag set, then if (time - stime) excessive, reset emerch(x); reset busy;
2. Endif; else set flag; set stime = time;
3. Endif; return;

Name: Norepm2
Admitted in: N NE SA CA
Input: -
Requires: Recunconfirm
If branch: Yes  
Condition: Flag? time excessive;
Effect: 1. If flag set, then if (time - stime) excessive, reset emerch(x); reset busy;
2. Endif; else set flag; set stime = time;
3. Endif; return;
Name: Norepm3
Admitted in: N NE SA CA
Input: -
Requires: Recunconfirm2
If branch: Yes
Condition: Flag? time excessive;
Effect: 1. If flag set, then if (time - stime) excessive, reset emerch(x); reset busy;
   2. Endif; else set flag; set stime = time;
   3. Endif; return;

Name: Passemerch
Admitted in: NE CA
Input: -
Requires: Buschek, emerch(1) flag = 9
If branch: Yes
Condition: Turning point here?
Effect: 1. If turning point here change lane; update vector from gate data; set flag= 10;
   2. If new ln# = 0, send fault-out to system;
   3. Endif; else call Callstop;
   4. Endif; return;

Name: Qallrec
Admitted in: NE CA
Input: -
Requires: Neset2
If branch: Yes
Condition: 4 flags? flag0? flag1? gate near?
Effect: 1. If flag0 is set, then if flag1 = 0, then
   2. If gate is near, then if all 4 flags are set,
   3. Send emerch_at_gate_x; set counter in Passemch; Flag = 9;
   4. Endif; else send unconfirm_emerch; reset busy; reset emerch(1);
   5. Endif; else if gate is near ;
   6. Else set flag1 = 0; reset flag0;
   7. Endif; else if gate is near set flag0;
   8. Endif; endif; return;

Name: Recadjahead
Admitted in: NE CA
Input: -
Requires: Buschek, emerch(1) and flag = 8
If branch: Yes
Condition: Adj_ahead_ok received?
Effect: 1. If adj_ahead_ok received, reset message flag; set flag in Qallrec;
   2. Endif; call recadjbehind; return;
Name: Recadjbehind
Admitted in: NE CA
Input: -
Requires: Recadjahead
If branch: Yes
Condition: Adj_behind_ok received?
Effect: 1. If adj_behind_ok received, reset message flag; set flag in Qallrec;
2. Endif; call recnextahead; return;

Name: Recackrqemch
Admitted in: NE CA
Input: -
Requires: Recnackrqemch
If branch: Yes
Condition: Replies received each position?
Effect: 1. If no ack_req_emerch messages received, set flag = 6; if speed of vehicle is low, call Sacall for adj and next;
2. Endif; else for each position (adj or next, ahead or behind):
a. If no relevant acks, set flag in Qallrec; if this is a behind position and speed of vehicle is low, call Sacall;
b. Endif; endif; send thanx_but_no to all irrelevant respondents;
3. Set flag = 7; endif; return;

Name: Recconemer
Admitted in: N NE SA CA
Input: -
Requires: Buschek emerch(x) (2 <= x <= 5) and flag = 8
If branch: Yes
Condition: Con-emerch received?
Effect: 1. If con-emerch received, then reset emerch(x); reset busy; revert to normal speed;
2. Else call Recunconfirm2;
3. Endif; return;

Name: Recposn
Admitted in: N NE SA CA
Input: Speed, posn, ln# of caller
Requires: Buschek, emerch(x), (2 <= x <= 5) and flag = 6
If branch: Yes
Condition: emerch-target received?
Effect: 1. If emerch-target received, store target and caller’s details; reset flag in Norepem2; flag = 7;
2. Else call Norepm1;
3. Endif; return;
Name: Rectnb 1
Admitted in: N NE SA CA
Input: -
Requires: Repwatel, flag = 5  
If branch: Yes  
Effect: 1. If thanx but no received, reset emerch(x), reset busy;  
2. Endif; return;

Name: Repwatel
Admitted in: N SA
Input: -
Requires: Buschek, emerch(1) and flag = 1,2,3,4,or 5.  
If branch: Yes  
Effect: 1. If flag = 5, call Recnackrqemch;  
2. Endif; flag = flag + 1; return;

Name: Repwatel
Admitted in: NE CA
Input: -
Requires: Buschek, emerch(x) (2 <= x <= 5) and flag = y (1 <= y <= 5)
If branch: Yes  
Effect: 1. If flag = 5, call Rectbnl; reset flag in Norepem 1;  
2. Endif; flag += 1; return;

Name: Recforsplitcall1
Admitted in: NE CA
Input: -
Requires: Voidemerch  
If branch: Yes  
Effect: 1. If for-split-called is received, reset message flag; send for-split-called to other partners in emerch; reset emerch(1); advise system; reset busy;  
2. Endif; return;

Name: Recforsplitcall2
Admitted in: NE CA
Input: -
Requires: Recnextbehind  
If branch: Yes  
Effect: 1. If for_split_called is received, reset message flag; send for-split-called to other partners in emerch; reset emerch(1); reset busy; advise system;  
2. Endif; call nest2; return;
Name: Recforsplitcal3  
Admitted in: N NE SA CA  
Input: -  
Requires: Forsplitcall  
If branch: Yes  
Condition: For-split called?  
Effect: 1. If for_split_called is received, reset message flag; reset busy; reset emerch(x);  
2. Endif; call recunconfirm; return;

Name: Recnextahead  
Admitted in: NE CA  
Input: -  
Requires: Recadjbehind  
If branch: Yes  
Condition: Next-ahead-ok received?  
Effect: 1. If next-ahead-ok received, reset message flag; set flag in Qallrec;  
2. Endif; call recnextbehind; return;

Name: Recnextbehind  
Admitted in: NE CA  
Input: -  
Requires: Recnextahead  
If branch: Yes  
Condition: Next-behind-ok received?  
Effect: 1. If next-behind-ok received, reset message flag; set flag in Qallrec;  
2. Endif; call recforsplitcal2; return;

Name: Recnackreqemch  
Admitted in: NE CA  
Input: -  
Requires: Repwate, flag = 5  
If branch: Yes  
Condition: Relevant nack_rq_emerch received?  
Effect: 1. If a relevant nack_req_emerch received, (relevant means that there is not another platoon behind the vehicle but ahead of the nacker), then  
2. If it is a nack because it system mode makes it impossible to reach this speed, reset own target speed;  
3. Endif; send thanx_but_no to all others who have sent messages; reset busy; reset emerch(1); reset all message flags;  
4. Else call Recackreqemch;  
5. Endif; return;
Name: Recreqemerchange  Fig: 6, 8
Admitted in: N NE SA CA
Input: Loc, ln# and flag from sender
Requires: Messrec and request-emerchange received
If branch: Yes  Condition: - Busy? NE or CA?
Effect:
1. Reset message flag;
2. If less than platoon spacing ahead of Loc of sender or if less than two spacings behind in lane to immediate right or next, then
3. If busy or if fault set or if mode = CA or NE or if mode = SA and stated speed of sender too high, call Sendnackrqemerch;
4. Else call Sendackrqemerch; if fault #6 or #7 is set,
5. Reroute transmission to system transmission and so for reception; set flag;
6. Endif; endif; endif; return;

Name: Recunconfirm  Fig: 6
Admitted in: N NE SA CA
Input: -
Requires: Recforsplcal3
If branch: Yes  Condition: Unconfirm_emerch received?
Effect:
1. If unconfirm-emerch received, reset busy; reset emerch(x);
2. Else call Norepem2;
3. Endif; return;

Name: Recuncontirm2  Fig: 6
Admitted in: N NE SA CA
Input: -
Requires: Reccomemer
If branch: Yes  Condition: Unconfirm_emerch received?
Effect:
1. If unconfirm-emerch received, reset busy; reset emerch(x);
2. Else call Norepem3;
3. Endif; return;

Name: SacalI  Fig: 6, Gen
Admitted in: NE CA
Input: Lane affected and Loc
Requires: Recackrqemch
If branch: No  Condition: -
Effect: From gate behind stated position to two ahead, lane stated to SA mode; return;
Name: Sendackrqemerch  
Admitted in: N SA  
Input: -  
Requires: Recreqemerch  
If branch: No  
Effect: 1. Set busy;  
   set emerch(2) (forward and in adjacent lane),  
   'emerch(3) (rearward and in adjacent lane),  
   emerch(4) (forward in next lane), or  
   emerch(5)(rearward in next lane);  
   2. Set flag = 1;  
   3. SendAck_request_emerchange with loc and ln# after x-lchrons for emerch(x);  
   return;

Name: Sendconemerch  
Admitted in: NE CA  
Input: -  
Requires: Neset3  
If branch: No  
Effect: Send confirm-emerch; reset busy; reset emerch(1); reset any lanes put into SA;  
   return;

Name: Sendnackrqemerch  
Admitted in: N NE SA CA  
Input: -  
Requires: Recreqemerch  
If branch: No  
Effect: Send nack_request_emerchng with loc, lane# and maxspeed; return;

Name: Sendreqemerch  
Admitted in: NE CA  
Input: -  
Requires: Neset1  
If branch: No  
Effect: Send Request-emerchange with lane, loc and maxspeed; return;

Name: Voidemerch  
Admitted in: NE CA  
Input: -  
Requires: Buschek, emerch(6), flag = 6  
If branch: Yes  
Effect: 1. If gate near, send emerch_to Void naming gate; flag = 9; reset flag in Callstop;  
   2. Else call Recforsplitcalll;  
   3. Endif; return;
A.6 PLATOON-LEADER’S PROBE (Figure A.7)

Name: Forprob  
Specification in: SUP

Name: Insight  
Admitted in: N NE SA CA  
Input: Distances of vehicles seen in probe (or null)
Requires: Forprob and flag = 1
If branch: Yes  
Condition: Vehicles visible?
Effect: 1. If any vehicle can be seen, select one most likely to be in same lane;
2. Send probe-l, including state vector and range of vehicle perceived (using corrected loc stored in Startprobe); set flag = 2;
3. Else send probe-nov, giving state vector; set flag = 9;
4. Endif; return;

Name: Nofault  
Admitted in: N NE SA CA  
Input: Message No-fault-l from system
Requires: Forprob, flag = 16, 17, 18
If branch: Yes  
Condition: Message No-fault-l rec’d?
Effect: 1. If message No-fault-l received, reset message flag; reset flag;
2. Else flag += 1;
3. Endif; return;

Name: Novehnorep  
Admitted in: N NE SA CA  
Input: -
Requires: Forprob, flag = 12
If branch: No  
Condition: -
Effect: Reset flag in forprob; return;

Name: Probeagain  
Admitted in: N NE SA CA  
Input: -
Requires: Forprob, flag = 5
If branch: No  
Condition: -
Effect: Send probe-:!, giving state vector; flag = 6; return;
Figure A.7. Forward Probe.
Name: Recackprobenov 1
Admitted in: N NE SA CA
Input: Location of sender
Requires: Forprob and flag = 9, 10, 11
If branch: Yes
Condition: Ack_probe_nov received?
Effect: 1. If ack_probe_nov received, work out distance and likely angle from stated location;
       2. If reply valid, flag = 15;
       3. Endif; endif; reset message flag; flag += 1; return;

Name: Recackprobenov2
Admitted in: N NE SA CA
Input: Location of sender
Requires: Forprob and flag = 13, 14, 15
If branch: Yes
Condition: Ack_probe_nov received?
Effect: 1. If ack_probe_nov received, work out distance and likely angle from stated location;
       2. If reply valid, flag = 20;
       3. Endif; endif; reset message flag; flag -= 1; return;

Name: Recackprobe1
Admitted in: N NE SA CA
Input: Location of sender
Requires: Forprob, and flag = 2,3,4
If branch: Yes
Condition: Ack_probe_1 received?
Effect: 1. If ack_probe_1 received, work out distance and likely angle from stated location;
       2. If reply valid, reset flag;
       3. Endif; endif; reset message flag; flag += 1; return;

Name: Recackprobe2
Admitted in: N NE SA CA
Input: Location of sender
Requires: Forprob, and flag = 6,7
If branch: Yes
Condition: Ack_probe_2 received?
Effect: 1. If ackgrobe 2 received, work out distance and likely angle from stated location;
       2. If reply valid (ie within 30 m or so), set flag = - 1;
       3. Endif; endif; reset message flag; flag += 1; return;
Name: Recackprobe2a
Admitted in: N NE SA CA
Input: Location of sender
Requires: Forprob, and flag = 8
If branch: Yes
Effect: 1. If ack_probe_2 received, work out distance and likely angle from stated location;
   2. If reply valid (ie within 30 m or so), reset message flag; reset flag;
   3. Else if only one vehicle visible, send Fault1_1_veh to system; reset message flag;
      flag = 16;
   4. Else flag = 19;
   5. Endif; endif; else if only one vehicle visible, send Fault1_1_yeh to system;
      flag= 16;
   6. Else flag = 19;
   7. Endif; endif; return;

Name: Recprobe 1
Admitted in: N NE SA CA
Input: -
Requires: Messrec
If branch: Yes
Effect: 1. If probe-1 received,
   2. Set flag in Forprob = 20;
   3. Else reset message flag;
   4. Endif; return;

Name: Recprobe2
Admitted in: N NE SA CA
Input: -
Requires: Messrec
If branch: Yes
Effect: 1. If probe-2 received
   2. Set flag in Forprob = 21;
   3. Else reset message flag;
   4. Endif; return;

Name: Recprobenov
Admitted in: N NE SA CA
Input: -
Requires: Messrec
If branch: Yes
Effect: 1. If probe-nov received,
   2. Set flag in Forprob = 22;
   3. Else reset message flag;
   4. Endif; return;

Fig: 7, 8
Condition: Ack_probe_2 received?

Condition: Probe-1 received?

Condition: Probe-2 received?

Condition: Probe-nov received?
Name: Sendackprobenov
Admitted in: N NE SA CA
Input: Loc of sender, stated range
Requires: Forprob, flag = 22
If branch: Yes
Condition: Within 1.5*platoon spacing of sender?

Effect: 1. If within range (1.5 times platoon spacing, say) of sender,
   2. Send Ack_probe_nov;
   3. Endif; reset flag; reset message flag; return;

Name: Sendackprobe 1
Admitted in: N NE SA CA
Input: Loc of sender, stated range
Requires: Forprob, flag = 20
If branch: Yes
Condition: At location specified

Effect: 1. If tolerably near (10 metres say) to specified range from sender,
   2. Send Ack_probe_1;
   3. Endif; reset flag; reset message flag; return;

Name: Sendackprobe2
Admitted in: N NE SA CA
Input: Loc of sender, stated range
Requires: Forprob, flag = 21
If branch: Yes
Condition: At location specified

Effect: 1. If approximately (25 metres say) at specified range from sender,
   2. Send Ack_probe_2;
   3. Endif; reset flag; reset message flag; return;

Name: Setfault1
Admitted in: N NE SA CA
Input: -
Requires: Forprob, flag = 19
If branch: No
Condition: -
Effect: Set fault #1; reset flag; return;

Name: StartProbe
Admitted in: N NE SA CA
Input: Loc of gate (gate is near)
Requires: Mersplit (10)
If branch: Yes
Condition: Pos? Busy? Fault #1 or #3 set?

Effect: 1. If busy ;
   2. Else if fault #1 or #3 set ;
   3. Else if pos = 1, then store any correction to loc as revealed by gate reading; set flag in Forprob = 1;
   4. Endif; endif; endif; return;
A.7 RECEIPT OF MESSAGES (Figure A.8)

Name: Buslead2
Specification in: Forced-Split (5)  Fig: 5, 8

Name: Buslead3
Specification in: For-split (5)  Fig: 5

Name: Cacall
Specification in: Gen  Fig: 8

Name: Crashstopcall
Specification in: Gen  Fig: 8

Name: Cutspeed
Specification in: Gen  Fig: 8

Name: Lastbusy
Specification in: For-split (5)  Fig: 5, 8

Name: Leadforsplit 1
Specification in: Forced-split (5)  Fig: 5

Name: Leadforsplit2
Specification in: Forced-split (5)  Fig: 5, 8

Name: Messrec
Specification in: SUP  Fig: 1, 8

Name: Necall
Specification in: Gen  Fig: 8

Name: Norcall
Specification in: Gen  Fig: 8

Name: Passon
Admitted in: N NE SA CA
Input: Message Request-merge rec’d, pos = last > 1
Requires: Messrec
If branch: Yes  Condition: Same lane as requester?
Effect: 1. If in same lane as requester, pass message up platoon;
2. Endif; return;

A-73
Figure A. 8. Messages.
(Messages responded to are in ovals.)
Figure A.8a. Messages (continued).

(Messages responded to are in ovals.)

A-75
Name: Recaheadb
*Specification in:* For-split’ (5)

Name: Recackreqmerbeh
*Specification in:* Merge(2)

Name: Recakrqssbeh
*Specification in:* Split (3)

Name: Recbehinda
*Specification in:* For-split (5)

Name: Reccantgo
*Specification in:* Joinquit (12)

Name: Recfault1probe
*Specification in:* Faults (9)

Name: Recinv_spl
*Specification in:* Split (3)

Name: Recprobel
*Specification in:* Forqrob (7)

Name: Recprobe2
*Specification in:* Forprob (7)

Name: Recprobenov
*Specification in:* Forqrob (7)

Name: Recreqsplit
*Specification in:* Split (3)

Name: Recreqmer3set
*Specification in:* Merge (2)

Name: Recreqchngln
*Specification in:* Change-lane (4)

Name: Recreqemerchange
*Specification in:* Emer-change (6)

Fig: 5, 8

Fig: 2, 16

Fig: 3, 8

Fig: 5, 8

Fig: 8, 12

Fig: 8, 9

Fig: 3, 8

Fig: 7, 8

Fig: 7, 8

Fig: 7, 8

Fig: 3, 8

Fig: 2, 8

Fig: 4, 8

Fig: 6, 8
Name: Recreq_mer  
*Specification in:* Merge (2)  

Name: Recreqsplchl  
*Specification in:* Split (3)  

Name: Recreqsplchlbeh  
*Specification in:* Split (3)  

Name: Sacall2  
*Specification in:* Gen  

Name: Sendackreqchln  
*Specification in:* Change-lane (4)  

Name: Sendackrqememerch  
*Specification in:* Emer-change (6)  

Name: Sendackreqmer  
*Specification in:* Merge (2)  

Name: Sendackreqs  
*Specification in:* Split (3)  

Name: Sendonf  
*Specification in:* For-split (5)  

Name: Sendackreqchln  
*Specification in:* Change-lane (4)  

Name: Sendackreqmer  
*Specification in:* Merge (2)  

Name: Sendackreqs  
*Specification in:* Split (3)  

Name: Sendackrqememerch  
*Specification in:* Emer-change (6)  

Name: Stopcall  
*Specification in:* Gen  

Fig: 2, 8  

Fig: 3, 8  

Fig: 3, 8  

Fig: 8  

Fig: 4, 8  

Fig: 6, 8  

Fig: 6, 8  

Fig: 5, 8  

Fig: 4, 8  

Fig: 2, 8  

Fig: 3, 8  

Fig: 6, 8  

Fig: 8  

A-77
Name: (Intfalt)  
Admitted in: N NE SA CA  
Input: MON routines  
Requires: Fault detected by internal monitor  
If branch: Yes  
Condition: Fault $#1?#11?$  
Effect: 1. Sets fault flag $#1-#11$ as appropriate;  
2. If fault $#1$, reset counter in Qtwofaults;  
3. Endif; else if fault 11, and speed $< c$ target, and Maystop flag reset, set Mystp flag;  
4. Endif;  

Note This routine is strictly part of MON
A.8 NEW FAULTS (Figure A.9)

Name: Newfault  Fig: 1, 9
Specification in: Sup

Name: Nextturn  Fig: 9
Admitted in: N NE SA CA
Input: -
Requires: Reclocishere, no message received
If branch: No  Condition: -
Effect: Decrement counter in Qtwofaults; return;

Name: Qtwofaults  Fig: 9
Admitted in: N NE SA CA
Input: Fault flags
Requires: Newfault, and some faults present
If branch: Yes  Condition: Proscribed or 3+ faults?
Effect: 1. If more than two faults (etc) present, call Sendstopit;
  2. Endif; if one of pairs of faults, modes and flags listed below present, call Sendstopit;
  3. Endif; if fault #1 is present, then if counter is set,
  4. Call Reclocishere;
  5. Else call Sendfaultlprobe;
  6. Endif; return;

Note
The following pairs of faults, etc. can induce a hazard, and therefore should induce Stopit.

   #1 with #2, #3, or #10
   #6 with #8
   #7 with #9
   #10 and CA mode
   Maystop flag and #1 or #3

Name: Recfaultlprobe  Fig: 8, 9
Admitted in: N NE SA CA
Input: Message fault-lqrobe received, with loc
Requires: Messrec
If branch: Yes  Condition: Same lane? In range? Pos?
Effect: 1. If pos = last, then
  2. If ln# and loc indicate that vehicle is in same lane as sender and not more than two platoon spacings ahead, send loc is here (includes loc);
  3. Endif; endif; return;
Figure A.9. Fault Flags.
Name: Reclocishere
Admitted in: N NE SA CA
Input: -
Requires: Qtwofaults, fault #1 and counter set
If branch: Yes
Condition: Loc is here received?
Effect: 1. If loc is here received, call Setspeed, passing loc of sender;
2. Else call nextturn;
3. Endif; return;

Name: Recmaystop
Admitted in: N NE SA CA
Input: -
Requires: Message Maystop from system
If branch: No
Condition: -
Effect: Sets relevant flag in Newfalt; return;

Note Maystop is sent when there is evidence of congestion.

Name: Recsetfault
Admitted in: N NE SA CA
Input: Message Setfault from system
Requires: Fault detected by system
If branch: Yes
Condition: Fault #1,#11?
Effect: 1. Sets fault flag #1,#1 1 as appropriate;
2. If fault #1, reset counter in Qtwofaults;
3. Endif; else if fault #11, and speed << target, and Maystop flag not set, set Maystop flag;
4. Endif; return;

Name: Setspeed
Admitted in: N NE SA CFA
Input: Loc and speed of sender of loc is here
Requires: Reclocishere
If branch: No
Condition: -
Effect: Set maxspeed to maintain 1.5*(normal platoon spacing), using normal longitudinal control algorithm;

Name: Sendfaultlprobe
Admitted in: N NE SA CA
Input: -
Requires: Qtwofaults, fault #1 present, counter reset
If branch: No
Condition: -
Effect: Send fault-lqrobe with loc and ln#; set counter in Qtwofaults = 4; return;
Name: Sendistop  
Admitted in: N NE SA CA 
Input: - 
Requires: Slowspeed and speed = 0, pos = 1

If branch: No 
Condition: - 
Effect: Vehicle has stopped behind vehicle ahead. No maystop sent Breakdown ahead? 
1. If no contact, send message I-stop to system, with loc of vehicle ahead; 
2. Else send collision; 
3. Endif; return;

Name: Slowspeed  
Admitted in: N NE SA CA 
Input: Speed reading, distance to next vehicle 
Requires: Newfalt

If branch: Yes 
Condition: Pos? 
Effect: 1. If pos = 1, send message slow-speed to system; (this includes speed and distance to veh ahead) [System may respond with Maystop, a degraded mode or a setfault] 
2. If speed = 0; call SendIstop; 
3. Endif; endif; return;

Name: Sendstopit  
Admitted in: N NE SA CA 
Input: - 
Requires: Qtwofaults

If branch: Yes 
Condition: Fault #8? 
Effect: 1. Vehicle has a dangerous combination of two faults or 3 or more. Maxspeed = 0; 
2. If fault #8, send message I-stop to gate, indicating loc and ln# and fault-in-me; (System will send Stopcall, inducing Stop mode) 
3. Else send I-stop (same message) to system; 
4. Endif; return;
A.9 NEW ACTIONS (Figure A.10)

Name: Actcalled
Specification in: SUP (1)

Name: Buslead 1
Specification in: Forsplit (5)

Name: Buslead4
Specification in: Forsplit (5)

Name: Callchanglane
Specification in: Change-lane (4)

Name: Callmerge
Specification in: Merge (2)

Name: Callsplit
Specification in: Split (3)

Name: Emerchcall
Specification in: Emer_change (6)

Name: Fault4or5
Admitted in: N NE SA CA
Input: -
Requires: Faultset, fault #4 or fault #5 present
If branch: Yes
Effect: 1. If pos = last call Forspla;
2. Else call Sendaheada;
3. Endif; return;

Condition: Pos?

Name: Forspla
Specification in: Forsplit (5)

Name: Forsplll
Specification in: Forsplit (5)

Name: Fs 1 set
Specification in: Forsplit (5)

Name: Fs33set
Specification in: Forsplit (5)

Fig: 1, 10, 12
Fig: 5, 10, 11
Fig: 5, 10
Fig: 4, 10
Fig: 2, 10
Fig: 3, 10
Fig: 6, 10
Fig: 10
Fig: 5, 10
Fig: 5, 10
Fig: 5, 10, 11
Fig: 5, 10, 11
Figure A.10. Action Calls.
Name: Invitesplit  
**Specification in:** Split (3)  

Name: Linkmess  
**Admitted in:** N NE SA CA  
**Input:** Messages from Link Control  
**Requires:** Faultset  
**If branch:** ‘Yes  

**Condition:** Any relevant messages?  
Free agent? Busy?

**Effect:**  
1. If busy, ;  
2. Else if Ln# = 0, then if message indicates enter; call Callchng_in;  
3. Else call Takeback;  
4. *Endif; endif;* if ln# = 1 and message indicates exit at next gate and  
   ownsize = 1, call Callchng_in;  
5. Else if message indicates exit at passed gate send passed-exit;  
6. *Endif; endif;* if (time-stime) < 0, stime = time;  
7. *Endif; endif;* if (time - stime) > 15 sec (say), then  
8. If ownsize = 1, then if message indicates change of n lanes to some loc = loc0  
   and loc > loc0 - n*X,  
9. Call Callchng_in;  
10. *Endif; else* if pos = 1 or pos = ownsize, then if message indicates change of n  
    lanes to some loc = loc0 and loc > loc0 - n*X - Y,  
11. Call Callsplit;  
12. *Endif; else* if message indicates change of n lanes to some loc = loc0 and  
    loc > loc0 - n*X - 2*Y,  
13. Call Callsplit;  
14. *Endif; endif;* else call Mersplit;  
15. *Endif; endif;* return;

*Note* Recording and processing of link messages is beyond scope of this specification. X is  
distance taken to change lanes. It is set by Link Control on entry (will be a function of flow):  
Y is distance to split - set in same way.
Name: Mersplit
Admitted in: N NE SA CA
Input: -
Requires: Linkmess and not busy
If branch: Yes
Condition: Pos = 1? Ownsize? Flag?
Time?
Effect: 1. If pos = 1, then if flag set, then if (time - stime) excessive (30 sec?),
2. Set stime = time; if ownsize c = MO and forward probe indicates vehicle
in range,
3. Reset flag; Call Callmerge;
4. Endif; If ownsize > = SO, reset flag; call Invite-split for vehicle with pos
= ownsize/2;
5. Endif; endif; else set flag; set stime = time; call Startprobe;
6. Endif;endif; return;

Note MO - Maximum size at which platoon will seek merge - and SO - Minimum size at
which platoon will split - are constants set by Link Control.

Name: Neset 1
Specification in: Emerch_change (6)

Name: Reqsplit
Specification in: Split (3)

Name: Reqchngln
Specification in: Change-lane (4)

Name: Sendaheada
Specification in: Forsplit (5)

Name: Sendbehindb
Specification in: Forsplit (5)

Name: Sendbehindl
Specification in: Forsplit (5)

Name: Sendreqemerch
Specification in: Emer_change (6)

Name: Startprobe
Specification in: Forprob (7)

Name: Takeback
Specification in: Join and Quit (12)
A.10 CONTROL DATA (Figure A.11)

Name: Buslead1
Specification in: Forsplit (5)

Name: Controlmess
Admitted in: N NE SA CA
Input: Flags 1 (control data) and 2 (ack_cont) from Contdat
Requires: Contdat
If branch: Yes
Condition: Flags
Effect: 1. If flag 1 is set, reset flag 1; set counter in Nocontmess to M; call Newvector;
   2. Else call Nocontmess;
   3. Call Recackcont (passing flags 1 and 2 as input);
   4. Return;

Name: Contdat
Specification in: SUP (1)

Name: Fs 1 set
Specification in: Forsplit (5)

Name: Fs2set
Specification in: Forsplit (5)

Name: Fs3set
Specification in: Forsplit (5)

Name: Newvector
Admitted in: N NE SA CA
Input: Control message
Requires: Controlmess
If branch: Yes
Condition: Change of leader?
Effect: 1. If message gives leader or ownsize different from state vector, then if busy,
   2. If merge(1) or merge(3) is set, reset it; reset busy; update state vector;
   3. Else if split(2) is set;
   4. Else if flag A is set, decrement A-counter; if A-counter = 0, then
   5. Reset flag A; set A counter to M; Message odd-change to system; reset busy;
      update state vector;
   6. Endif; else set flag A; set A-counter to M;
   7. Endif;endif;endif;endif;else if flag B is set, decrement B-counter; if B-counter = 0, then
   8. Reset flag B; set B-counter to M; update state vector;
   9. Endif; else set flag B; set B-counter to M;
10. Endif; else reset flag A; set A counter to M; reset flag B; set B-counter to M;
11. Endif; return;
Figure A.11. Control Data.
Name: Nocontmess
Admitted in: N NE SA CA
Input: -
Requires: Contmess
If branch: Yes
Condition: Flags? Counter?
Effect:
1. If Newlead flag is set; send control data as leader; decrement counter; if counter = 0, then
2. Reset newlead flag; set counter = M; call Sendbehindbl;
3. Endif; else decrement counter; if counter = 0, then
4. Send control data as leader; set newlead flag; set counter = M;
5. Endif; endif; return;

Name: Recackcont
Admitted in: N NE SA CA
Input: Flags
Requires: Controlmess
If branch: Yes
Condition: Flag? Counter?
Effect:
1. If flag 2 passed is set, reset flag X; set counter to 2;
2. Else if flag 1 is set, decrement counter X; if counter = 0, then
3. Reset flag X; set counter = 2; call Sendaheada;
4. Endif; endif; return;

Name: Sendaheada
Specification in: Forsplit (5)

Name: Sendbehindb 1
Specification in: Forsplit (5)

Name: Sendunmergel
Specification in: Forsplit (5)
A.11 JOINING AND LEAVING (Figure A.12)

Name: Actcalled

Specification in: SUP (1)

Name: Ask

Admitted in: Entry lane
Input: Driver request
Requires: None!
If branch: Yes
Condition: Destination specified?
Effect:
1. If destination specified, send request-enter to system;
   2. While true,
      a. examine message flags; if any set,
      b. if it is Are_you_fitted, reset flag; break;
      c. endif; else reset flag;
      d. endif;
   3. Endwhile; Go to Monsys;
   4. Else display to driver “No destination stated”;
   5. Endif; stop;

Name: Manualok

Admitted in: Exit lane
Input: -
Requires: Takeback
If branch: Yes
Condition: Message Manual-OK received?
Effect:
1. If message manual-ok received,
   2. Message to driver - “have a good day!“; autocontrol off; stop;
   3. Else call TooFar;
   4. Endif; return;

Name: Monsys

Admitted in: Entry lane
Input: Are_you_fitted rec’d
Requires: Ask
If branch: Yes
Condition: Controls fit?
Effect:
1. If MON indicates controls fit, send Im_OK; go to Waitentry;
   2. Else send Im_off; go to Nogol;
   3. Endif;
Figure A.12. Joining and Leaving.

A-91
Name: Nogol  
Admitted in: Entry lane  
Input: -  
Requires: Monsys - controls unfit  
If branch: No  
Effect: Display to driver “Self monitor indicates fault(S) #x, get vehicle repaired”; stop;

Name: Nogo  
Admitted in: Entry lane  
Input: Text with message no-entry  
Requires: Waitentry - no-entry received  
If branch: No  
Effect: Display to driver “Entry refused - (message text); stop;

Note Reason may be in vehicle - license outdated, carrying external load, etc or within system - downstream section closed, destination in other direction, etc.

Name: Reccantgo  
Admitted in: Entry lane  
Input: Message cant_go received, with gate ID  
Requires: Messrec or Waitentry  
If branch: No  
Effect: Alter link data to indicate exit at this gate; set flag in takeback; send I-quit; return;

Name: Takeback  
Admitted in: N NE SA CA  
Input: -  
Requires: Linkmess (10) or Reccantgo (13)  
If branch: Yes  
Condition: Flag  
Effect: 1. If flag set, display to Driver: “Sorry - entry barred”;  
2. Endif; Display to driver “Please take control”; call Manualok; (driver can send Im_ready}  
3. Return;

Name: Takeover  
Admitted in: Entry lane  
Input: Message You-are-in received  
Requires: Waitentry  
If branch: No  
Condition: -  
Effect: 1. Set link data included in message; set state vector (from message); reset flag in Takeback;  
2. Go to Actcalled;
**Name:** Toofar  
**Admitted in:** Exit lane  
**Input:**  
**Requires:** Manualok  
**If branch:** Yes  
**Effect:**  
1. If any message received,  
2. On message,  
   Manual-ok: break;  
   Inch-on:  
   End-of-lane:  
   {default}:  
3. Endif; return;  

---

**Name:** Waitentry  
**Admitted in:** Entry lane  
**Input:**  
**Requires:** Monsys  
**If branch:** Yes  
**Effect:**  
1. While true,  
   a. examine message flags; if any set,  
   b. on flag,  
      You-are-in: go to Takeover;  
      No-entry (with text): reset flag; go to Nogo2;  
      Cantgo: reset flag; go to Reccantgo;  
      (default): reset flag;  
   c. endif;  
2. Endwhile;
A.12 SYSTEM

Note: This description does not explain how the platoon-level roadside system works. It will in fact cycle steadily, and respond to many messages from and to many vehicles. Here we consider the interaction with each vehicle separately.

The interaction takes the form of receipt of a message and a response, which may or may not require consultation of stored data.

All the following messages are received and retransmitted with the marker indicating fault #6 or #7 in the vehicle making the lane-change. Each is simply retransmitted.

Message: Request-emer-change (Sendreqemerch, fig 6)
Respond with: Request_emer_change;

Message: Ack_request_emer_change (Sendackrqemch, fig 6)
Respond with: Ack_request_emer_change;

Message: Nack_request_emer_change (Sendnackrlemch, fig 6)
Respond with: Nack_request_emer_change;

Message: Confirm_emer_change (Sendconemerch, fig 6)
Respond with: Confirm-emer-change;

Message: Thanx_but_no (Recackrqemch, recnackrqemch, fig 6)
Respond with: Thanx_but_no;

Message: Unconfirm_emer_change (Callstop, fig 6)
Respond with: Unconfirm_emer_change;

Message: Emerch_to_void (Voidemerch, fig 6)
Respond with: Emerch_to_void; set turning-point;

Message: Emer_change_at_gate_X (Qallrec, fig 6)
Respond with: Emer_change_at_gate_X; set turning-point;

Message: Adj_ahead_ok etc (Inposn, fig 6)
Respond with: Adj_ahead_ok;
The following messages cause the reactions stated in the descriptions referred to:

**Messages:**
Nesetl (Nesetl, fig 6) No-entry mode, if fault #6 or #7 for next two sections
Neset2 (Neset2, fig 6) No entry mode extended as gate passed
Neset3 (Neset3, fig 6) No-entry mode transformed following lane-change, and SA mode also removed.

**Message:** SAcall (Sacall, fig 6)
Not restricted to a particular fault. Calls SA mode to enable a safe emer-change.

**Message:** Fault-xqresent (Actcalled, fig 10)
**Respond with:** Add vehicle to list of faulty vehicles; if a vehicle remains on this list too long, advise system operators.

**Message:** Fault-out (Passemch, fig 6)
**Respond with:** Remove vehicle from list; if system operators advised of its presence, advise again;

**Message:** Fault_1_veh (Forprob, fig 7).
**Respond with:** If another vehicle with fault #4 or #5 is present send No-fault-l; else no action;

Gate messages.

The following messages do not provoke a transmitted message but what is said below:

**Messages:**
Confirm_split_change_lane (in transmitted form)(Recconsplit, fig 4)
Confirm-dropt (Sendcondropt, fig 4)
Confirm-decel (Sendcondecel, fig4)
Change-to-void (Voidch, fig 4)
Emerch_to_void (Voidemurch, fig 6)
Emerch_at_gate_x (Qallrec, fig 6)

**Respond with:**
1. While 8 secs pass,
2. While giving-side VPD has not yet been occupied or is occupied,
3. If receiving side VPD is not occupied, activate turning point;
4. Else deactivate turning-point;
5. Endif; endwhile; deactivate turning point;
6. Endwhile;

Note. In the case of an exit gate, the receiving-side VPD extends downstream.
Entry and Exit Messages.

Message: Request-entry. (Askin, fig 12)
Respond with: Are_you_fitted to vehicle. Record existence, destination.

Message: Im_OK (MONsys, fig 12)
Respond with: 1. If licence valid and destination achievable, send Urin.
2. Else send No-entry, with reason as text; delete record of vehicle;
3. Endif;

Message: Im_off (MONsys, fig 12)
Respond with: -. Delete record of vehicle.

Message: [Vehicle has passed last on-gate]
Respond with: [Localized message] Cant-go

Message: I-quit (Reccantgo, fig 12)
Respond with: -. Delete record of vehicle

Message: [Vehicle almost at end of TL after last on-gate](Toofar, fig 12)
Respond with: [Localized message] Inch-on

Message: [Vehicle at end of TL] (Toofar, fig 12)
Respond with: [Localised message] End-of-lane

Message: Im_ready (Takeback, fig 12)
Respond with: 1. While brakes are being applied;,
2. Endwhile; send Manualok; delete vehicle from record;

Mode Changes.

All these messages are followed by advice to system operators, who alone can reverse their effects.

Messages: Crashstopbehind (Qfalt 1, fig 5), also unnamed messages indicating fence breach, or intrusion of platoon crush on to detector at gate.
Respond with: Crashstopcall (see Gen) for vehicles in platoon calling, plus two following sections; Cacall for following (N) sections; SAcall2 for parallel sections on lanes adjacent to CA section; advise system supervisors;
Messages: Callstop (Callstop, fig 6), I-stop (SendIstop, fig. 9)
Collision (SendIstop, fig.9), Stopit (Sendstopit, fig.9)
Respond with: Stopcall (see Gen) for two sections behind caller;
Cacall for following (N) sections; SACall2 for parallel sections on lanes adjacent to CA sections; advise system supervisors;

Message: Slowspeed (Slowspeed, fig. 9),
Respond with: 1. If other reports in same area send May-stop to affected lanes; advise supervisors;
2. Else store info;
3. Endif;

Message: Unnamed message indicating slow-moving or stationary vehicle on exit lane, downstream of primary detector VPD.
Respond with: SACall2 for exit lane

Suspect vehicles list.

Messages:
Norepchll (Norepchll, fig 4)
Norepchi2 (Norepchi2, fig 4)
Norepchi3 (Norepchi3, fig 4)
Norepchi4 (Norepchi4, fig 4)
Norepchi5 (Norepchi5, fig 4)
No_tp_1 (Syscall1, fig 4)
No_tp_2 (Syscall2, fig 4)
Change-messed-up (Syscall3, fig 4)
Too-long (Toolong, figs 2,3)
Respond with: 1. If there is a recently reported faulty vehicle in the same area, ;
2. Else if one of the vehicles concerned has been involved in such incidents before,
3. Advise supervisors; Send Setfalt#ll;
4. Else if gate (not for Too-long) has been involved before,
5. Close gate; advise supervisors;
6. Else add vehicles involved to suspect vehicles list;
7. Endif; endif; endif; return:
## A.13 LIST OF MESSAGES

The list of messages below gives the references to figures (6 = Figure A.6, etc), in which messages are referred to.

- **up** = message transmitted forward in platoon
- **Down** = message transmitted rearward in platoon
- **out** = message transmitted between platoons
- **Ex** = message transmitted to system
- **In** = message transmitted by system
- **=** message is not addressed to particular vehicle

<table>
<thead>
<tr>
<th>Message</th>
<th>Figure</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledge-control-data</td>
<td>11</td>
<td>Up</td>
</tr>
<tr>
<td>Ack_request_change_lane</td>
<td>4</td>
<td>out</td>
</tr>
<tr>
<td>Ack_request_emerch</td>
<td>6</td>
<td>out</td>
</tr>
<tr>
<td>Ack_request_merge</td>
<td>2</td>
<td>Out,Down</td>
</tr>
<tr>
<td>Ack_request_split</td>
<td>3</td>
<td>Down</td>
</tr>
<tr>
<td>Adj_ahead_OK</td>
<td>6</td>
<td>out</td>
</tr>
<tr>
<td>Adj_beh_OK</td>
<td>6</td>
<td>out</td>
</tr>
<tr>
<td>Cacall</td>
<td>8</td>
<td>In</td>
</tr>
<tr>
<td>Callstop</td>
<td>5, 6</td>
<td>Ex</td>
</tr>
<tr>
<td>Cant_go</td>
<td>12</td>
<td>In</td>
</tr>
<tr>
<td>Change-mixed-up</td>
<td>4</td>
<td>Ex</td>
</tr>
<tr>
<td>Change-to-void</td>
<td>4</td>
<td>Out,Ex</td>
</tr>
<tr>
<td>Collision</td>
<td>9</td>
<td>out</td>
</tr>
<tr>
<td>Confirm-change-lane</td>
<td>4</td>
<td>out</td>
</tr>
<tr>
<td>Confirm-decelerate</td>
<td>4</td>
<td>Out,Ex</td>
</tr>
<tr>
<td>Confirm-dropt</td>
<td>4</td>
<td>Out,Ex</td>
</tr>
<tr>
<td>Confirm-emerch</td>
<td>6</td>
<td>Out,Ex</td>
</tr>
<tr>
<td>Confirm-merge</td>
<td>2</td>
<td>Out,Up</td>
</tr>
<tr>
<td>Confirm-split</td>
<td>3</td>
<td>out</td>
</tr>
<tr>
<td>Confirm_split_change_lane</td>
<td>4</td>
<td>Out,Ex</td>
</tr>
<tr>
<td>Control data</td>
<td>11</td>
<td>Down</td>
</tr>
<tr>
<td>Crashstop</td>
<td>Many</td>
<td>Ex,In</td>
</tr>
<tr>
<td>Crashstopcall</td>
<td>8</td>
<td>In</td>
</tr>
<tr>
<td>Cutspeed</td>
<td>8</td>
<td>In</td>
</tr>
<tr>
<td>Emerch_at_gate_x</td>
<td>6</td>
<td>Out,Ex</td>
</tr>
<tr>
<td>Emerch_target</td>
<td>6</td>
<td>Ex</td>
</tr>
<tr>
<td>Emerch_to_void</td>
<td>6</td>
<td>Out,Ex</td>
</tr>
<tr>
<td>Fault-lgrobe</td>
<td>9</td>
<td>out*</td>
</tr>
<tr>
<td>Fault-xgresent</td>
<td>9</td>
<td>Ex</td>
</tr>
<tr>
<td>Forced-split-called</td>
<td>4,5</td>
<td>out</td>
</tr>
<tr>
<td>Message</td>
<td>Figure</td>
<td>Link</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>Forsplit_b_(N)_f</td>
<td>5,7</td>
<td>Up</td>
</tr>
<tr>
<td>Forsplit_b_(N)_r</td>
<td>5,7</td>
<td>Down</td>
</tr>
<tr>
<td>Forsplit_over</td>
<td>7</td>
<td>Down</td>
</tr>
<tr>
<td>Gatepassed</td>
<td>6</td>
<td>Ex</td>
</tr>
<tr>
<td>Im_off</td>
<td>12</td>
<td>Ex</td>
</tr>
<tr>
<td>Im_OK</td>
<td>12</td>
<td>Ex</td>
</tr>
<tr>
<td>Im_ready</td>
<td>12</td>
<td>Ex</td>
</tr>
<tr>
<td>Inch-on</td>
<td>12</td>
<td>Ex</td>
</tr>
<tr>
<td>Invite-split</td>
<td>3</td>
<td>Down</td>
</tr>
<tr>
<td>I-quit</td>
<td>12</td>
<td>Ex</td>
</tr>
<tr>
<td>I-stop</td>
<td>9</td>
<td>Ex</td>
</tr>
<tr>
<td>Maystop</td>
<td>8</td>
<td>In</td>
</tr>
<tr>
<td>Merge_3_set</td>
<td>2</td>
<td>Down</td>
</tr>
<tr>
<td>Nack_request_change_lane</td>
<td>4</td>
<td>out</td>
</tr>
<tr>
<td>Nack_request_emerch</td>
<td>6</td>
<td>out</td>
</tr>
<tr>
<td>Nack_request_merge</td>
<td>2</td>
<td>Out, Down</td>
</tr>
<tr>
<td>Nack_request_split</td>
<td>3</td>
<td>Down</td>
</tr>
<tr>
<td>Necall</td>
<td>8</td>
<td>In</td>
</tr>
<tr>
<td>Neset1</td>
<td>6</td>
<td>Ex</td>
</tr>
<tr>
<td>Neset2</td>
<td>6</td>
<td>Ex</td>
</tr>
<tr>
<td>Neset3</td>
<td>6</td>
<td>Ex</td>
</tr>
<tr>
<td>Next-ahead-OK</td>
<td>6</td>
<td>out</td>
</tr>
<tr>
<td>Next_beh_OK</td>
<td>6</td>
<td>out</td>
</tr>
<tr>
<td>No-fault-1</td>
<td>7</td>
<td>In</td>
</tr>
<tr>
<td>Norcall</td>
<td>8</td>
<td>In</td>
</tr>
<tr>
<td>Norepchl 1</td>
<td>4</td>
<td>Out,Ex</td>
</tr>
<tr>
<td>Norepchl2</td>
<td>4</td>
<td>Out,Ex</td>
</tr>
<tr>
<td>Norepchl3</td>
<td>4</td>
<td>Out,Ex</td>
</tr>
<tr>
<td>Norepchl4</td>
<td>4</td>
<td>Out,Ex</td>
</tr>
<tr>
<td>Norepchl5</td>
<td>4</td>
<td>Out,Ex</td>
</tr>
<tr>
<td>No_tip_1</td>
<td>4</td>
<td>Out,Ex</td>
</tr>
<tr>
<td>No_tip_2</td>
<td>4</td>
<td>Out,Ex</td>
</tr>
<tr>
<td>Probe-1</td>
<td>7</td>
<td>out*</td>
</tr>
<tr>
<td>Probe-2</td>
<td>7</td>
<td>out*</td>
</tr>
<tr>
<td>Probe-nov</td>
<td>7</td>
<td>out*</td>
</tr>
<tr>
<td>Request-change-lane</td>
<td>4</td>
<td>out*</td>
</tr>
<tr>
<td>Request-decelerate</td>
<td>4</td>
<td>out</td>
</tr>
<tr>
<td>Request-emerch</td>
<td>6</td>
<td>out*</td>
</tr>
<tr>
<td>Request-entry</td>
<td>12</td>
<td>Ex</td>
</tr>
<tr>
<td>Request-merge</td>
<td>2</td>
<td>Out*, Up</td>
</tr>
<tr>
<td>Request-split</td>
<td>3</td>
<td>Up</td>
</tr>
<tr>
<td>Request-split-change-lane</td>
<td>3, 4</td>
<td>Down</td>
</tr>
</tbody>
</table>

A-99
<table>
<thead>
<tr>
<th>Message</th>
<th>Figure</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sacall</td>
<td>6</td>
<td>Ex</td>
</tr>
<tr>
<td>Sacall2</td>
<td>8</td>
<td>In</td>
</tr>
<tr>
<td>Slowspeed</td>
<td>9</td>
<td>Ex</td>
</tr>
<tr>
<td>Split-over</td>
<td>3</td>
<td>Down</td>
</tr>
<tr>
<td>s topcall</td>
<td>8</td>
<td>In</td>
</tr>
<tr>
<td>Stopit</td>
<td>9</td>
<td>Ex</td>
</tr>
<tr>
<td>Thanx_but_no</td>
<td>496</td>
<td>out</td>
</tr>
<tr>
<td>Toofar</td>
<td>12</td>
<td>In</td>
</tr>
<tr>
<td>Toolong</td>
<td>2,3,4</td>
<td>Ex</td>
</tr>
<tr>
<td>Unconfirm-change-lane</td>
<td>4</td>
<td>out</td>
</tr>
<tr>
<td>Unconfirm_emerch</td>
<td>6</td>
<td>out</td>
</tr>
<tr>
<td>Unmerge</td>
<td>2,5</td>
<td>out</td>
</tr>
</tbody>
</table>