

**Using Group Communication to Support
Inter-Vehicle Coordination**

Eoin O’Gorman B. Eng.

A dissertation submitted to the University of Dublin,
in partial fulfilment of the requirements for the degree of
Master of Science in Computer Science.

2002

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed: _____

Eoin O’Gorman

16th September, 2002.

Permission to lend and/or copy

I agree that Trinity College Library may lend or copy this dissertation upon request.

Signed: _____

Eoin O’Gorman

16th September, 2002.

Acknowledgments

I wish to thank the following individuals for their contribution:

My supervisor, Dr. Vinny Cahill, for his time, help and advice throughout the year, which I appreciate very much, thank you;

My Mum, Dad, Granny, my brothers and especially my sister, Clodagh, for “putting me up”, always with a smile, for an entire year.

Finally, I would also like to thank my girlfriend Jennie to all her support.

Abstract

Recent advancements in wireless communication technology and portable computing are opening up exciting possibilities for the future of mobile network applications. One obvious domain for such mobile network applications is in vehicle traffic scenarios, where vehicle-mounted mobile hosts, with wireless communication ability, form an ad hoc network. With the aim of improving transportation systems, through the reduction of road accidents, inter-vehicle coordination applications are being built upon these vehicle-based ad hoc networks.

This dissertation is concerned with investigating the use of group communication to support inter-vehicle coordination applications in ad hoc networks. Specifically, one particular case study of inter-vehicle coordination is examined: the 4 Way Stop. A 4 Way Stop is a common road junction, with four approaching roads, of equal importance, meeting at a single point. The problem of a 4 Way Stop is in determining which vehicle should be allowed to cross the junction at any one time.

This dissertation identifies the requirements on group communication to successfully support a 4 Way Stop application. Requirements for optional group communication application implementations: primary component implementation and partitionable membership implementation are both listed. Moving from the specific to the general, the examination of a 4 Way Stop application provides valuable insights into the general domain of inter-vehicle coordination.

Table of Contents

Abbreviations	1
Chapter 1: Introduction	2
1.1 The Group Communication Paradigm	3
1.1.1 Applications of Group Communication	3
1.1.2 Group Communication Terminology	4
1.2 Inter-vehicle Coordination	6
1.3 Related Technologies	6
1.3.1 Global Positioning System	7
1.3.2 Obstacle Detection	7
1.4 Dissertation Organisation	9
Chapter 2: State of the Art	10
2.1 Group Communication Services	10
2.1.1 ISIS, Cornell University	11
2.1.2 Horus, Cornell University	12
2.1.3 Transis, Hebrew University of Jerusalem	14
2.1.4 Totem, University of California	14
2.2 Location Aware Group Communication Research	15
2.2.1 “Safe Distance”	15
2.2.2 “Proximity Groups”	17
2.2.3 Architecture for location-aware group members	17
2.3 Wireless Traffic Applications	18
2.3.1 “FleetNet – Internet on the Road”	18
2.3.2 DriveBy InfoFueling™	20
2.3.3 Traffic jam detection using wireless technology	21
2.3.4 CarNet	22

2.3.5 GPS-based message broadcasting	23
Chapter 3: The 4 Way Stop Problem	24
3.1 Junction Layout	24
3.2 The Problem	25
3.3 4WS Automated Vehicle Model	26
3.4 Application-Level Traffic Scenarios	28
Chapter 4: Mapping 4 Way Stop Problem to Group Communication	37
4.1 Membership Interest	37
4.2 Group Communication Primitives	37
4.2.1 Ordering of Group Communication Primitives	39
4.3 Shared Data Structure	41
4.3.1 Maintenance of Shared Data Structure using Primitives	47
4.3.2 Right of Way Algorithm	53
Chapter 5: Group Communication Requirements	56
Related Work	56
Primary component and partitionable membership services	56
Operating Environment: Fault Model and Assumptions	57
Proposed System Architecture	58
Notation	59
Primary Component Membership Required Properties	61
Property 1: “Self Inclusion”	62
Property 2: “Local Monotonicity”	62
Property 3: “Initial View Event”	63
Property 4: “Primary Component Membership”	63
Property 5: “Delivery Integrity”	64
Property 6: “No Duplication”	64
Property 7: “Same View Delivery”	65
Property 8: “Virtual Synchrony”	65
Property 9: “Safe Indication Prefix”	66
Property 10: “Strong Total Order”	67
Partitionable Membership Required Properties	69
Property 1: “Self Inclusion”	70
Property 2: “Local Monotonicity”	70
Property 3: “Initial View Event”	71

Property 4: “Delivery Integrity”	72
Property 5: “No Duplication”	72
Property 6: “Same View Delivery”	73
Property 7: “Virtual Synchrony”	73
Property 8: “Transitional Set”	74
Property 9: “Safe Indication Prefix”	75
Property 10: “Weak Total Order”	76
Chapter 6: Conclusion	79
Future Work	81
References	83

List of Figures and Tables

Figure 2.1: Group communication microprotocols envisaged as Lego blocks [20].	13
Figure 2.2: Example of safe distance [41].	16
Figure 2.3: Layered Architecture for Group Communication [43].	18
Figure 2.4: High level FleetNet infrastructure and communication methods [44].	19
Figure 2.5: Example DriveBy InfoFueling scenarios [46].	21
Figure 2.6: Example of detecting the traffic jam size [19].	22
Figure 3.1: 4WS Junction (aerial view).	25
Figure 3.2: Example 4WS scenario, illustrating rule of right of way granting.	26
Figure 3.3: An empty junction.	28
Figure 3.4a: A single vehicle approaching an empty junction.	29
Figure 3.4b: Multiple vehicles approaching an empty junction “simultaneously”.	29
Figure 3.5a: Vehicle approaches a non-empty junc. along an empty approach road.	29
Figure 3.5b: Vehicle approaches a non-empty junc. along an non-empty approach road.	29
Figure 3.6: Vehicle seizes right of way (with the <i>intention</i> of crossing the junction).	30
Figure 3.7a: A vehicle seized right of way and left the junction subsequently empty.	30
Figure 3.7b: A vehicle seized right of way and left the junction subsequently non-empty.	30
Figure 3.8a: A vehicle, before seizing RoW, leaves the junction subsequently empty.	31
Figure 3.8b: A vehicle, before seizing RoW, leaves the junction subsequently non-empty.	31
Figure 3.9a: A vehicle stalls at the front of an approach road.	32
Figure 3.9b: A vehicle stalls on an approach road, but not the front.	32
Figure 3.9c: A vehicle stalls on the junction.	32
Figure 3.10a: A local vehicle obstruction at the front of an approach road.	33
Figure 3.10b: A local vehicle obstruction on an approach road, but not at the front.	33
Figure 3.10c: A local vehicle obstruction on the junction.	33
Figure 3.11: Vehicle relinquishes right of way before entering on to the junction.	34

Figure 3.12a: A vehicle reports a blockage at the front of an approach road..	35
Figure 3.12b: A vehicle reports a blockage along an approach road, but not the front.	35
Figure 3.12c: A vehicle reports a blockage on the junction.	35
Figure 3.13a: Vehicle reports a removal of a blockage on the junction.	36
Figure 3.13b: Vehicle reports a removal blockage of a blockage along an approach road.	36
Figure 4.1: Example Traffic Scenario.	43
Figure 4.2: Element 1: “Membership Queue”.	44
Figure 4.3: Element 2: “Right of Way History”.	45
Figure 4.4: Local vehicle blocking all vehicles on approach road 3.	46
Figure 4.5: Element 3: “Blockage Status”.	46
Figure 4.6: Local vehicle blocking subset of vehicles on approach 3.	46
Figure 4.7: Example data structure elements after group is created.	47
Figure 4.8: Example “join” traffic scenario.	48
Figure 4.9: Membership queue update after join.	49
Figure 4.10: Before and after membership queue for an “expected leave”.	49
Figure 4.11: Before and after right of way history for an “expected leave”.	50
Figure 4.12: Before and after membership queue for “seize right of way”.	50
Figure 4.13: Before and after membership queue for “relinquish right of way”.	51
Figure: 4.14: Before and after blockage status for “report junction blocked”.	51
Figure: 4.15: Before and after blockage status for “report approach blocked”.	52
Figure: 4.16: Before and after blockage status for removal of blockage on junc.	52
Figure 4.17: Before and after blockage status for removal of blockage on approach.	53
Figure 4.18: Right of Way algorithm.	55
Figure 5.1: Proposed System Architecture.	59
Figure 5.2: External actions of the GCS [23].	60
Table 5.1: External actions of the GCS.	60
Table 5.2: Shorthand Predicates.	61
Table 5.3: Summary of primary component requirements.	68
Figure 5.3a: A traffic scenario before partition.	69
Figure 5.3b: A traffic scenario after partition.	69
Table 5.4: Summary of partitionable membership service requirements.	77

Abbreviations

4WS	4 Way Stop
AHN	Ad Hoc Network
EVS	Extended Virtual Synchrony
GCS	Group Communication Service
GLS	Grid Location Service
GMP	Group Membership Protocol
GMS	Group Membership Service
GPS	Global Positioning System
ITS	Intelligent Transportation Society
IVC	Inter-Vehicle Communication
IVI	Intelligent Vehicle Initiative
LAN	Local Area Network

Chapter 1: Introduction

Recent advancements in wireless communication technology and portable computing are opening up exciting possibilities for the future of mobile network applications. Through the combination of wireless communication ability and mobile devices, applications of self-organising, infrastructure-free, mobile networks are being developed in many domains: educational, industrial, commercial and military. Mobile devices form a mobile ad hoc network when they wirelessly communicate with other nearby wireless devices without the support of fixed infrastructure. The mobility of nodes means that an ad hoc network's topology changes frequently and dynamically. With the absence of fixed infrastructure to support the monitoring of these network topology changes, mobile nodes must themselves track changes in a decentralised manner.

We propose investigating the use of group communication to support tracking the network topology changes in an ad hoc network. Specifically, we propose investigating the use of group communication in ad hoc networks formed by mobile vehicles with wireless communication ability. These vehicles will form an ad hoc network with the purpose of coordinating their respective actions and driving manoeuvres. Interest in so-called inter-vehicle coordination is growing. Groups such as the Intelligent Transportation Society (ITS), the Intelligent Vehicles Initiative (IVI), vehicle manufactures and, indeed, academics are researching inter-vehicle coordination with a common goal of improving transportation systems through the reduction of road accidents, the improvement of traffic congestion, the availability of traffic information and the reduction of traffic-related environmental pollution.

As an example of inter-vehicle coordination, built upon group communication in ad hoc networks, we study the problem of the 4 Way Stop. This specific problem, which is fully

described in chapter three, was used as a case study of inter-vehicle coordination to identify the requirements on group communication to support inter-vehicle coordination in mobile ad hoc networks. The identifications of these requirements is the objective of this dissertation. Through an investigation on a specific inter-coordination application, we will make observations on such applications in general.

1.1 The Group Communication Paradigm

Group communication is a powerful paradigm for creating distributed services and highly resilient, fault-tolerant applications [1, 2]. Group communication is a method of providing point to multi-point communication due to the arrangement of processes into groups. According to a principle of mutual cooperation group member processes interact and communicate to achieve a common objective. The group communication paradigm is an important and widely used technique to achieve reliability, fault-tolerance and high availability in distributed computing [3, 4].

What makes the group communication paradigm novel is that an entire set of processes can be considered as a single logical connection. This is accomplished through the notion of *group abstraction* which allows processes to be grouped together into multicast groups each of which are assigned group identifications [1, 2]. Messages to these multicast groups are addressed simply using the group's identification, which is assigned when the group is created. Depending on the strength of the associated multicast semantics, a multicast message addressed to a group is guaranteed to reach all currently connected, functional members. This means

1.1.1 Applications of Group Communication

Many applications have been built using the group communication paradigm. A classic group communication application is data replication using a variant of the state machine/active replication approach [5, 6, 7]. Other classical group communication applications include load balancing [8, 9], distributed transactions and database replication [10, 11], resource allocation [12, 13], and highly available servers [14].

More recently, the group communication paradigm has been applied to collaborative computing applications such as distributed whiteboards [15], audio and video conferencing [16, 17] and distance learning [18].

Group communication has also been applied to vehicle traffic scenarios. Traffic jam detection and emergency braking are example of group communication used in traffic applications [19]. This dissertation will look to using group communication in coordinating automated vehicles. Member vehicles will coordinate their actions through the principle of mutual assistance, where vehicles keep each other informed of the state of the local traffic environment by location-enriched message passing. Group communication based traffic applications are faced with the challenge of operating in a wireless environment. Problems such as node mobility and message loss are obstacles to such applications. However, the current trend towards safer vehicles through automation motivates the development of such wireless group communication applications.

1.1.2 Group Communication Terminology

This section describes the key ideas and terminology in group communication. Group Communication Services (GCSs) are middleware systems that hide underlying network inconsistencies from distributed applications and provide the applications with consistent views of a distributed execution [4, 20, 21, 22]. GCSs are particularly useful for providing applications with reliable point to multi-point communication.

The idea of *group abstractions* is used by GCSs to group distributed processes into multicast groups. All multicast group members can then be addressed by applications using only their group id, which is assigned to multicast groups upon creation. The usefulness of the group abstraction is that different members of a group each have a consistent view of all group communication. In general, these semantics are achieved using *group membership* and *reliable multicast*.

Group membership is concerned with maintaining a view of the distributed processes that comprise a group. Membership of a group is dynamic over the lifetime of a group due to processes joining, leaving, failing or disconnecting. A Group Membership Service (GMS) is responsible for tracking the changes to the group and reporting these changes to the members of the group in the form of *group views* [2]. Group views, consist of a group

view identification and a list of the current members of the group. The test of a group membership service is in delivering each connected member of the group consistent views of the group i.e. each member is to be delivered a list of currently connected members, so that members know their possible communication partners.

Reliable multicast is concerned with providing applications with the ability to reliably send messages to all members of a specific group. Different GCSs may provide numerous reliable multicast semantics: different reliable multicast protocols have different properties. For example, different message ordering properties could be specified. Example multicast semantics include: FIFO, Reliable FIFO, Causal, Reliable Causal and Totally Ordered [23]. However, most reliable multicast semantics can at least guarantee the property of virtual synchrony.

Virtual Synchrony semantics specify how group communication is synchronized with group view delivery [24]. The essence of the virtual synchrony principle is that it guarantees that membership changes within a process group are observed in the same order by all the group members that remain connected. Virtual synchrony guarantees that every two processes that observe the same two consecutive group membership changes, receive the same set of group multicast messages between the two group membership changes. Virtual synchrony places no requirements on the order of delivery or receipt of the set of multicast messages. A separate protocol could be built on top of virtual synchrony which could ensure a specific ordering of messages at all group members. With regards consistent distributed application execution, virtual synchrony means that, at group view changes, members transitioning from one consistent group view to the next have identical contexts. This means that members can act upon received messages in a consistent manner, as if synchronised.

To illustrate the virtual synchrony principle, consider the following example. Consider a group of processes that is initially comprised of processes {A, B, C} but changes to {A, B, D}. Processes A and B will initially receive a configuration message indicating that the group consists of A, B and C. A and B will then receive the exact same set of regular communication messages, followed by the configuration message indicating the group now consists of A, B and D. Process C, however, after the first configuration message may receive any superset or subset of the set of messages received by A and B. The result

of this is that A and B, who participate in identical, consecutive group views have the same contexts when the second group view configuration message arrives and hence can act on messages in a consistent manner.

1.2 Inter-vehicle Coordination

This dissertation is concerned with investigating the use of the group communication paradigm in the area of inter-vehicle coordination. Inter-vehicle coordination involves vehicles communicating and interacting, using inter-vehicle communication (IVC), so as to organize their respective movement and driving manoeuvres. Current examples of inter-vehicle coordination applications include organising vehicles into tightly spaced platoons for efficient traffic flow, cooperative lane changing and overtaking [25, 26, 27]. Our research specifically looks at coordinating automated vehicles crossing a 4 Way Stop (4WS) junction. Based solely on message passing, location-aware vehicles must unambiguously determine which vehicle is allowed to cross the junction at any one time. This coordination will be built upon group communication, which in turn will be built upon communication in an ad hoc network. This research was motivated by the need to explore and investigate the requirements on group communication to coordinate vehicles operating as mobile nodes in an ad hoc network.

Group communication offers many useful tools for implementing such an inter-vehicle coordination application, however, we will investigate the requirements on current GCSs to fully coordinate automated vehicles at a 4WS junction in an ad hoc network. Ad hoc network characteristics such as message loss, frequent disconnections, network partitions and unpredictable node mobility make reliable coordination difficult. Also, inter-vehicle coordination using group communication must accurately reflect real world vehicle situations i.e. the group communication paradigm must respect actual physical traffic constraints such as vehicle queuing order.

1.3 Related Technologies

Although our research is primarily concerned with inter-vehicle coordination using group communication in ad hoc networks, other technologies are assumed. This section details the two main technologies assumed by the remainder of this dissertation: the Global Positioning System (GPS) and Vehicle Obstacle Detection.

1.3.1 Global Positioning System

A fundamental assumption of the 4WS protocol, and other similar safety critical inter-vehicle protocols, is that a vehicle is aware of its global position. Numerous vehicles currently use the American Government's Global Positioning System (GPS) [28, 29] as part of their navigation system. Established in the early 1990's, GPS is the a system designed to give a relatively accurate (to within 10m) position on Earth, anytime, anywhere and in any weather. 24 GPS satellites orbit the Earth, each transmitting signals which can be detected by a GPS receiver. Using a minimum of three received satellite signals, a GPS receiver is able to provide a global position.

Currently, GPS can only offer positioning accuracy to within 10m, however, improvements in GPS accuracy are expected within the next few years. In [30], Rogers and Schroedl predict that future GPS vehicle systems will be accurate to one meter and better by 2005. This improvement is expected to be achieved using differential corrections. Improvements in GPS accuracy coupled with improvements in the accuracy of digital road maps using probe vehicles [31] will enable precision vehicle safety applications, such as the 4WS application, to be developed.

1.3.2 Obstacle Detection

A primary challenge in developing intelligent autonomous vehicles is reliable obstacle detection. Roadways present an unknown and dynamic environment with real-time constraints. Also, the high speeds of vehicle travel require a system to detect objects at long distances. There has been a great amount of research devoted to the obstacle detection problem in intelligent vehicles.

One method of alerting vehicles of obstacles on roadways is the deployment of sensors on the roadside. These sensors regularly scan the roadway for the presence of obstacles and broadcast information to vehicles in the locality. A significant problem with this solution is the expensive deployment of these roadside sensors. A more suitable solution would be for the vehicles to detect the presence of obstacles themselves, and hence free the detection of obstacles from a reliance on roadside support.

A variety of competing methods for obstacle detection have been proposed for example optical flow [32, 33], stereo vision [34, 35], radar [36] and laser range-based [37, 38].

One of the most effective methods of obstacle detection are laser range scanners, or ladars, which have been used for many years for obstacle detection. Laser scanners operate by sweeping a laser across a scene and at each angle, measuring the range and returned intensity. The laser provides an elevation map of the scanned area. This elevation map is scanned for discontinuities which indicate obstacles. Radar is another excellent means of detecting obstacles. One implementation of radar obstacle detection is capable of detecting vehicles at distances of up to 200m [39]

For the purpose of this dissertation, we require vehicles to have the ability to detect the presence of an obstacle within maximum braking distance in front of the vehicle. The chosen method of obstacle detection is unimportant, assuming, however, that the obstacle detection sensors are contained within the vehicle and not on the roadside. This dissertation is concerned with infrastructure-free ad hoc networks and, as such, it would not be rational to constrain a final implementation to rely on roadside obstacle detection sensors.

1.4 Dissertation Organisation

This dissertation is organised into six chapters as follows:

In chapter two we detail the state of the art concerned with our research.

In chapter three we introduce the 4 Way Stop problem, which is the specific example of inter-vehicle coordination with which this dissertation is concerned.

In chapter four we detail the initial steps towards developing a 4 Way Stop application using group communication.

Chapter five is concerned with detailing the requirements on group communication to successfully implement a 4 Way Stop application in both primary component and partitionable network environments.

Finally, chapter six concludes the discussion.

Chapter 2: State of the Art

This dissertation is concerned with inter-vehicle coordination built upon group communication, where group members are aware of their global location using GPS. As a result, our literature review will examine current and previous research and industrial development in the areas of group communication, location aware group communication and wireless applications in inter-vehicle scenarios.

The examination of group communication research was motivated by the need to understand how existing Group Communication Services (GCSs) have been implemented and identify novel features of their implementations. Location aware group communication is concerned with group communication where group members have some understanding of their physical position. This is the case with inter-vehicle coordination built upon group communication and therefore it was advantageous to identify possible use of location information by mobile nodes. A study of current wireless traffic applications was also carried out. This study was carried out with the intention of identifying similar traffic applications and classifying the methods of implementation used.

2.1 Group Communication Services

In this section, we detail research carried out into Group Communication Services. To understand features of the well-known GCSs we studied ISIS [2, 3, 4], Horus [2, 20], Transis [22, 40] and Totem [21]. These GCSs are middleware systems which provide distributed applications with group communication semantics, thus increasing reliability and availability. Also, the GCS's provide applications with reliable point to multi-point communication thus enabling the applications to implement reliable data replication. This study of existing GCSs was carried out so as to understand the basics of group

communication including group membership, group membership tracking, system architecture and multicast semantics.

2.1.1 ISIS, Cornell University

ISIS [2, 3, 4] was first developed at Cornell University in 1987 as a group communication toolkit with the aim of providing fault tolerance in distributed systems. The ISIS system implements a collection of techniques for building software for distributed systems that performs well, is robust despite both hardware and software crashes, and exploits parallelism. As the ISIS Group Membership Service (GMS) was the original group membership service, it was of particular importance to this dissertation to study this GMS so as to understand the basics and original notions of group membership. The following is a review of how group membership in ISIS is tracked.

The ISIS GMS maintains the group membership list on behalf of the group members. The ISIS GMS assumes a dynamic membership environment in which the group membership is represented by strictly ordered group views. ISIS assumes a model in which possible member processes must contact the GMS in order to join the group. The GMS itself must be a highly available entity, which itself may have a dynamic membership necessitated by GMS member process failures.

As stated, the primary role of the ISIS GMS is to track membership of processes within the system, however, in order to accomplish this the GMS must first track its own membership. This is accomplished by the Group Membership Protocol (GMP). The GMP handles the addition and deletion of GMS member processes and replicates the GMS membership list amongst the GMS members. Member failures are detected through members monitoring each others' status using a ping-timeout system.

In order to track the membership of the system, the ISIS GMS offers three operations to system members: join, leave and monitor. The join and leave operations are self-explanatory. The monitor operation is used by a process if it suspects another process of having failed. The provision of this operation is necessary because the GMS must handle all failure notification i.e. member processes are not allowed to assume another process has failed. The use of the monitor operation by the ISIS GMS tracks membership of the entire group in the background.

As a primary component service, the ISIS GMS may fail to progress under certain conditions. In order to create a new group view i , the GMS must contain a *majority* of group members from group view $i-1$.

As the first and most famous GCS, ISIS provides an invaluable investigation into the history and original implementation of group communication.

Much of the work of this dissertation follows on from the investigation and analysis of the ISIS system. ISIS being the original and most famous GCS provides much of the understanding of group membership and group membership monitoring. However, it must be noted that ISIS assumes an environment in which message loss is not possible. Such an assumption cannot be made by inter-vehicle coordination applications, which will have to operate in asynchronous environments.

2.1.2 Horus, Cornell University

The Horus middleware system [2, 20] offers application developers a flexible group communication model. Horus was designed to facilitate adding group communication semantics such reliability, fault-tolerance and security to already existing applications without significant changes to the application. The flexibility of Horus is due to the fact that Horus embeds group communication support into a modular systems architecture. Before Horus, GCSs tended to offer a “flat” architecture: in that the offered API was fixed and closely matched to exact group communication primitives, regardless of the specific requirements of the distributed application using the GCS middleware. Horus, on the other hand, offers a configurable group communication architecture, meaning that Horus’ API is dynamic depending on the application’s requirements.

The configurable group communication architecture is achieved by a protocol block abstraction. These protocol blocks can be looked upon as Lego blocks, where the Horus system is a box containing the Lego blocks (see figure 2.1 taken from [20]). Each block offers a unique group communication microprotocol, with standardised top and bottom interfaces. Combinations or stacks of blocks form macroprotocols, which contain the desired group communication properties of the distributed applications.

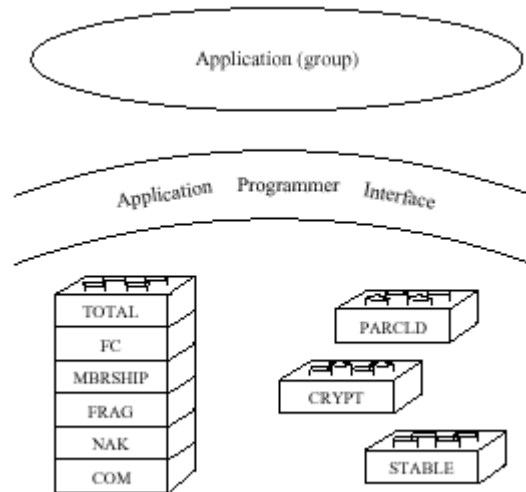


Figure 2.1: Group communication microprotocols envisaged as Lego blocks [20].

This modular architecture of group communication protocols, to support a distributed application, can be configured at runtime to match the requirements of the specific application. Horus' flexibility comes from the fact that its protocol stack can be optimized by including or excluding specific protocol stack blocks, or modules, depending on the specific application's requirements. For example, in a secure environment group communication need not be encrypted so Horus offers the application developer the option of removing the encryption/decryption functionality from the protocol stack. Other example microprotocol (block) functionalities include: encryption/decryption (CRYPT), provision of group membership list through use of virtual synchronous protocol (MBRSHIP), flow control (FC), totally ordered messaging (TOTAL) and detection of stable messages at all endpoints (STABLE).

Although novel, the architectural modularity of Horus is not relevant to the group communication functionality that will be offered by a 4 Way Stop application. Horus was designed to "slide under" existing distributed applications to offer optional and configurable group communication semantics. A 4 Way Stop application has specific group communication requirements, which will be detailed in later chapters. These requirements will be the sole requirements implemented by the group communication layer of a 4 way Stop application. To offer a dynamic set of configurable group communication semantics to a specific application, with its own specific group

communication requirements, would be overkill. The study of Horus was useful in that the Horus configurable architecture provided an insight into the fact that group communication semantics can be modularized, an insight which proved useful in identifying specific 4 Way Stop requirements.

2.1.3 Transis, Hebrew University of Jerusalem

Transis [22, 40] is a multicast communication middleware system that was designed to facilitate easier development of fault tolerant distributed applications over a network of machines. Transis supports high availability applications through the provision of reliable group communication and associated multicasts.

The primary design feature of Transis to support ease of application development was that it offers distributed application developers numerous multicast protocols with different ordering guarantees. Specifically, Transis offers: FIFO ordered, causally ordered, totally ordered, and safely delivered. As multicast semantics vary, so do the associated costs of the multicasts (e.g. message latency, number of rounds etc.), with FIFO ordered and causally ordered having the least associated cost. Application developers choose a multicast based on an application's requirements. For example, total ordering of messages, which incurs a high cost, means that all messages are delivered to all applications in the same order. Application developers would use such multicast semantics for data replication applications. Also, for example, causal ordering will guarantee that a response to a message will never be delivered to an application before the original message.

Of particular interest to this dissertation is Transis' observation: that different multicast semantics are required for different application requirements. Transis offers software developers a suite of multicast semantics. With regards this dissertation, the specific case study on inter-vehicle coordination too will require specific multicast semantics for its specific requirements, which shall be detailed in a later chapter.

2.1.4 Totem, University of California

Totem [21] is a group communication system designed to facilitate productive software development of reliable, fault-tolerant group applications involving replicated data in local area networks. This is accomplished by the Totem system's provision of totally

ordered multicast message delivery throughout the entire process group system. A protocol hierarchy delivers totally ordered messages to group members over a LAN or even over multiple LANs connected by gateways. The Totem system offers two reliable totally ordered multicast message delivery services: *agreed delivery* and *safe delivery*. Of particular use is safe delivery which guarantees that before a process delivers a message, it has determined that every other process in the current group view has received the message. Total ordering is achieved using a system of logical time stamps, token passing and message sequence numbers.

Totem is an example of the use of multicasting messages to achieve data replications in a network. Unlike this dissertation, Totem is only concerned with operation in a LAN. Operation in an ad hoc network poses serious network problems which will be detailed in a later chapter. However, the use of multicast semantics to replicate data will be of importance in the remainder of this dissertation.

2.2 Location Aware Group Communication Research

Research has been carried out into group communication with an emphasis on the mobile nodes being aware of their physical location. Likewise, this dissertation will be concerned location-aware hosts. Using GPS technology, vehicles approaching the 4WS junction will know their global location and, in conjunction with digital roadmaps, their position on the road/highway. The following section examines other research in the area of location aware mobile hosts, with the aim of understanding how mobile node location information has been used.

2.2.1 “Safe Distance”

In ad hoc networks unexpected node disconnection can be frequent and common, the network may even partition and never remerge. [41] uses node location information to decide if a node is to be admitted to or eliminated from a group. Node location and movement is monitored and announced disconnection is forced to ensure that nodes are eliminated from the group before an unexpected disconnection can occur. Node connectivity is mapped by a logical connective graph and not a physical connective graph. The former is a sub-graph of the latter. Both graphs share the same vertices (nodes), but the logical graph is missing some of the edges (links). The concept of *safe distance* (figure

2.2 based on [41]) is used to generate the logical connective graph. Taking node relative velocity, node transmission range and network latency, for a given communication task, into account a maximum safe distance is calculated for any two nodes. If the actual physical distance between the two nodes does not exceed this threshold distance then these two nodes can complete the communication task in question.

With regards figure 2.2a (based on [41]), mobile nodes a and b are within communication range, R . Both a and b may wish to form a group, however, under the concept of safe distance, they will not be allowed to do so. This is because, a and b may move out of communication range immediately after acknowledging membership of the group, resulting in messages, sent between them, being lost. As stated above, the solution of safe distance enforces nodes to be within a threshold distance from each other before membership of a group is established. In figure 2.2b, the threshold distance is shown as r , where:

$$r = R - 2v * (t + t')$$

R is the transmission range of mobile hosts;

t is the upper bound on message latency;

t' is the time required for a group level operation (group merge / split);

and, v is the maximum speed of a mobile host.

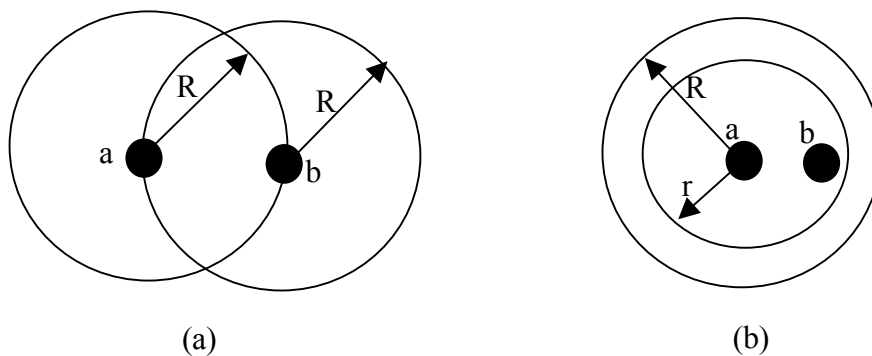


Figure 2.2: Example of safe distance [41].

Safe distance is an example of novel use of location information to predict mobile node disconnection. However, this novel design assumes that all inter-node communication is symmetric and that link failure cannot occur when nodes are within communication range. These assumptions can not be made for the 4WS protocol because message loss

and asymmetric communication are common in ad hoc networks. To ignore these issues in a safety critical protocol would not be acceptable.

2.2.2 “Proximity Groups”

In [42] Killijian, Cunningham, Meier, Mazare and Cahill introduce a new definition of *proximity groups*. According to their definition, a proximity group takes into account a mobile node’s location information and functional aspects before membership is allowed. Location alone does not warrant a node’s membership of a group. An example cited is a group of vehicles in the vicinity of traffic lights which are interested in receiving traffic light state changes. Clearly, this example of an absolute proximity group (i.e. a proximity group formed around an fixed geographic location) is concerned with both location information (area around traffic lights) and functional aspects (interest in traffic light changes). In order to be able to apply for group membership, a node must be within the geographical area corresponding to the group and must also be interested in the group. [42] also details a novel approach to coverage estimation and partition anticipation of member nodes.

The definition of absolute proximity groups is extended in this dissertation from the general to the specific. The remainder of the dissertation will be concerned with an example absolute proximity group comprised of vehicles in the vicinity of a 4WS junction (location aspect) and who are interested in gaining right of way at the junction (functional aspect).

2.2.3 Architecture for location-aware group members

Prakash and Baldoni [43] propose an architecture for group communication in mobile systems based on mobile nodes’ location awareness. Under this architecture, a group is composed of all nodes within a certain distance from the group co-ordinating node. The proposed architecture introduces two layers between the Application Layer and the Underlying Network Layer: the Proximity Layer and Group Membership Layer (figure 2.3, taken from [43]). The Proximity Layer uses MAC sub layer services to determine all mobile hosts within a certain distance from the original group creator. The Group Membership Layer, which is built upon the Proximity Layer, constructs group views based on a three phase protocol. This architecture, however, does not address the issues of network partitions and host disconnection. These issues are of paramount concern to any

protocol implementing a distributed inter-vehicle co-ordinating application where human life is at stake. What is important to this dissertation is the layered architecture and the modularisation of the various subsections of an group communication-based application.

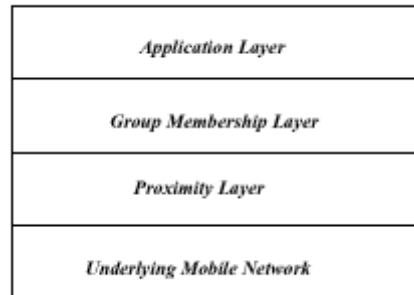


Figure 2.3: Layered Architecture for Group Communication [43].

2.3 Wireless Traffic Applications

As wireless networking is gaining increased importance within the communications industry, more and more applications for wireless technologies are being investigated. The automotive industry has proved to be an obvious area in which to develop such wireless applications. Academics and automotive industrialists alike are both researching novel uses of mobile technology in vehicles. Likewise, the automotive industry has proved an ideal area for development of civilian applications using GPS. As a result, we will examine research being carried out in the area of wireless traffic applications, specifically traffic applications using GPS technology.

2.3.1 “FleetNet – Internet on the Road”

The FleetNet project [44] was set-up by a consortium of six companies and three universities with the objective of developing an open-standard inter-vehicle communication platform to handle data exchange between vehicles and between roadside infrastructure and vehicles. The development of such a communication platform is fuelled by vehicles and drivers’ increased need for information on their current location. Information regarding traffic flow conditions, available local services and even entertainment information are all valid types of information which could be distributed by

future FleetNet systems. All results of the FleetNet project will be open, available international standards.

FleetNet assumes vehicle-vehicle and vehicle-infrastructure radio communication in an ad hoc network (figure 2.4 from [44]). FleetNet assumes the roll-out of permanently immobile nodes in the form of FleetNet roadside gateways which provide vehicles with access to the Internet. The FleetNet ad hoc network is not designed to be stand alone. Other nodes with the FleetNet network are mobile nodes (moving vehicles) and temporarily immobile nodes (parked vehicles), which together with the roadside infrastructure form the ad hoc network (although, the validity of such an “ad hoc network” could be questioned due to the presence of fixed infrastructure).

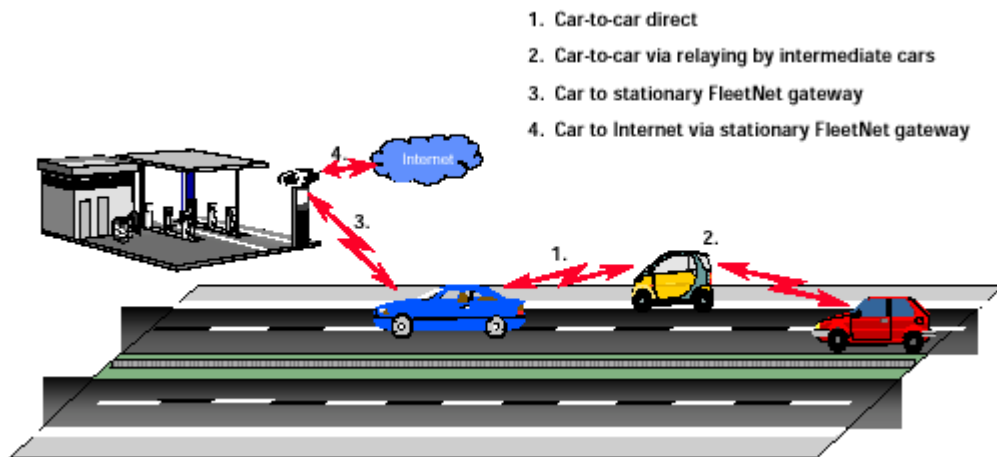


Figure 2.4: High level FleetNet infrastructure and communication methods [44].

The FleetNet project is attempting to address both technical and marketing issues of developing such a system. The main technological challenges facing the project are: the development of suitable communication protocols for use in such an ad hoc network environment; the choice of appropriate existing radio broadcasting hardware for standardisation; the integration of Internet availability into the ad hoc network environment; the interpretation of data and the presentation of interpretations to the driver in a safe manner. Marketing issues facing the FleetNet project are due to the fact that ad hoc network applications require a principle of mutual assistance in order to for the

application to function as specified. Therefore, a FleetNet system will require a high market penetration in order for meaningful applications to be provided.

FleetNet applications are designed with passenger comfort and safety in mind. [44] cites three FleetNet application classes: 1) cooperative driving assistance e.g. emergency braking; 2) decentralised floating car data applications e.g. traffic jam detection and 3) user communication and information services e.g. online games and chat.

The FleetNet project is concerned with inter-vehicle communication, however, unlike this dissertation, the FleetNet platform is being designed to cater for a general suite of applications which involve information distribution and integration to the Internet. We are concerned with a more specific application of reliably coordinating automated vehicles within a stand alone ad hoc network using group communication. Also, FleetNet assumes the use of fixed infrastructure for some of its intended applications, this dissertation makes no such assumption meaning cheaper rollout of services.

2.3.2 DriveBy InfoFueling™

DaimlerChrysler Research [45] are currently investigating a new high performance, low cost, scalable technology called “DriveBy InfoFueling” [46]. Similar to the suggested infrastructure of FleetNet, this technology involves providing an infrastructure of roadside wireless local area network access points. These network access points will allow passing road vehicles to download large volumes of data from the internet. An example scenario is shown in figure 2.5, taken from [46]. Research estimates that a vehicle passing an InfoFueling station at 60 MPH would be able to download many Megabytes of data in a matter of seconds. Some quoted in-vehicle applications for use in conjunction with DriveBy InfoFueling are: vehicle navigation, news provision (traffic, current affairs, stock market information etc.) and entertainment (radio, video, email etc.).

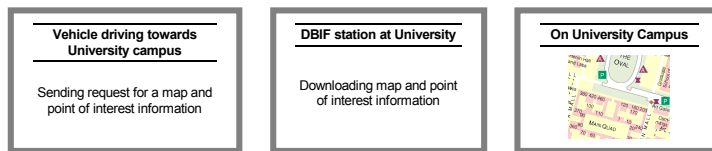


Figure 2.5: Example DriveBy InfoFueling scenarios [46].

Of particular interest to this dissertation is DriveBy InfoFueling’s contribution to vehicle navigation. DaimlerChrysler envisage that a vehicle passing a InfoFueling station will be able to download up to date local traffic information in the form of a digital road map. This digital road map will be used in conjunction with the vehicle’s GPS equipment to help the driver navigate through a strange city or town. Navigation using DriveBy InfoFueling shows the possibility of an innovative joining of wireless and GPS technologies in the area of traffic control. However, the necessity of fixed infrastructure in the new technology could prohibit its development and will increase associated costs. This dissertation will look at implementing a decentralised vehicle protocol without the need for fixed infrastructure yet involving both GPS and wireless technologies. Such an implementation of a traffic application would be cheaper to deploy and quicker to market.

2.3.3 Traffic jam detection using wireless technology

In [19] Breisemeister explores a new routing paradigm in the context of inter-vehicle communication in ad hoc networks. The routing paradigm implicitly addresses message destinations based on the network situation. Scoped, controlled flooding is used to reach destination nodes. Considering possible scaling problems, nodes only maintain a localized group membership list, which is comprised only of adjacent nodes within radio range. The sample inter-vehicle application built on this new paradigm detects the presence of a traffic jam and estimates the size of the jam (figure 2.6 from [19]). Using digital road maps and GPS equipment, slow vehicles share their location information with other

vehicles in order to distribute the traffic situation. Other vehicles can then detect and estimate the size of traffic jams.

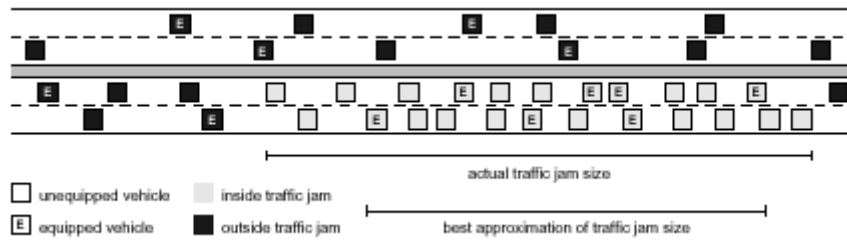


Figure 2.6: Example of detecting the traffic jam size [19].

This inter-vehicle system was designed to operate in sparsely connected ad hoc networks. Also, a system constraint was that 100% deployment was not required for the system to operate, however, traffic jam estimations may be inaccurate as a result. This system is an example of original use of a vehicle's location information in an ad hoc network environment. Similarly, the 4WS system will make clever use of node location information, yet an important difference must be noted: in order to operate correctly, 100% deployment of the 4 Way Sop system will be necessary. Also, the 4WS system will involve reaching distributed agreement as to which vehicle should receive right of way at the junction, whereas, Breisemeister's system is concerned with incomplete dissemination of information about the traffic situation and hence vehicles have an incomplete observation of the traffic situation.

2.3.4 CarNet

[47] describes CarNet, a new wireless network system for vehicles being developed at MIT. CarNet is designed around the authors' work on GLS (Grid Location Service). In GLS a node's location is distributed throughout the network and maintained in nodes which act as location servers for other nodes. Geographic information is used by a source node to forward packets to a destination node. Intermediate nodes make local decisions to forward the packet to geographically closer nodes. CarNet is an application of inter-vehicle communication in an ad hoc network, which will be used to develop other traffic applications which need to be geographically aware such as congestion monitoring, fleet tracking, resource location discovery and even inter-vehicle chat.

CarNet is designed to be a large scale, widely deployed inter-vehicle system. It shows how geographic location within an ad hoc network can be used to develop useful distributed applications. From this we can explore the idea of using geographic location and distributed consensus to develop a right of way system at a 4WS junction. However, geographical forwarding, in which only a subset of all possible nodes are reached, will be irrelevant in the 4WS system as *all* nodes will have to reach a distributed consensus as to which node should be granted right of way.

2.3.5 GPS-based message broadcasting

In [48] the authors propose a new wireless broadcast protocol specifically designed for inter-vehicle scenarios using GPS technology. The authors cite two important differences between Inter-Vehicle Communication (IVC) and standard ad hoc networks. Firstly, node speed in IVC is assumed to be faster than in average ad hoc networks. This has the effect of making vehicles travelling in opposite directions be only briefly connected. Secondly, node velocity is not random as vehicles travel along fixed roads. This observation is exploited by the new broadcast protocol in that message propagation need only occur in a single dimension i.e. the direction of the road and not in two or three dimensions as would be expected in standard ad hoc networks. The key to the new protocol is that a (re)sending node only selects a subset of neighbouring nodes to rebroadcast its message. The advantage of only selecting a subset of nodes to (re)broadcast the message is that bandwidth utilization is improved while maintaining node reachability. The choice of the subset of re-broadcasting nodes is based on nodes' GPS location. Although not directly related to the 4WS system, this new broadcast protocol again shows how GPS location information and inter-vehicle communication can be used in innovative ways.

With the state of the art identified, the next chapter looks at a specific case study of inter-vehicle coordination. The problems of coordinating vehicles in a 4 Way Stop are identified. This case study serves to provide an insight into inter-vehicle coordination applications and scenarios in general.

Chapter 3: The 4 Way Stop Problem

This chapter details the inter-vehicle coordination scenario that we investigated. The problem is concerned with a 4 Way Stop (4WS) junction. A 4WS is a common road junction, with four approaching roads, of equal importance, meeting at a single point.

In the following sections we detail the physical environment of a 4WS junction. The problem of the 4WS is explained. We also describe the vehicle model which is assumed to operate in the 4WS environment. Finally, all application 4WS scenarios are listed.

In the next chapter, chapter four, we detail steps towards implementing a 4WS application using group communication. It must be noted, before reading this chapter, that a core feature of a group communication implementation of the 4WS problem is the development of a suitable data structure, which will be shared among vehicles. The purpose of this data structure is to provide vehicles with shared knowledge so that they can independently determine which vehicle should have right of way at the junction.

3.1 Junction Layout

So as to understand the 4WS rules and the eventual shared data structure, it is first necessary to understand the physical environment in which a 4WS protocol will be expected to operate. As can be seen in figure 3.1, a 4WS junction consists of four approach roads, of equal importance, joining at a single point. For descriptive purposes, approach roads are labelled 1 to 4 in a clockwise manner. Also, it must be noted that this dissertation assumes that vehicles travel on the *left hand side* of the road.

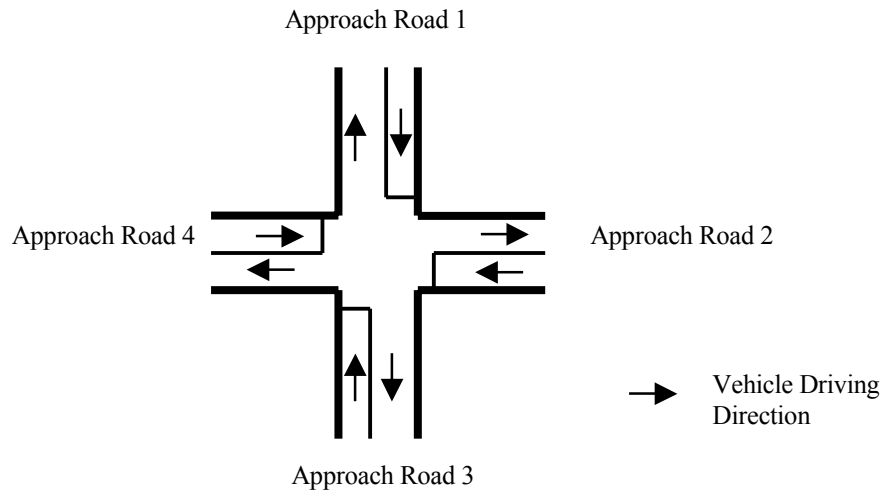


Figure 3.1: 4WS Junction (aerial view).

3.2 The Problem

The problem presented to automated vehicles approaching a 4WS junction is:

Which vehicle should have right of way at the junction?

The general rule which we assume for granting right of way at a 4WS junction is:

The vehicle longest awaiting right of way at the front of an unblocked 4WS approach will be granted right of way.

For example, in figure 3.2 an example traffic scenario is shown. Four vehicles (1, 2, 3 and 4) are queuing on four different 4WS approaches. Assume that vehicle 1 has been waiting, at the front of approach road 1, for right of way longer than any of the other three vehicles. Vehicle 2 has been queuing for the second longest period of time at the front of an approach, vehicle 3 the third longest period of time and vehicle 4 the most recent vehicle to queue at the front of an approach road. In such a scenario, the vehicle to be granted right of way first will be vehicle 1, followed by vehicle 2, then vehicle 3 and finally vehicle 4. This is due to the fact that vehicle 1 has been queuing at the front of an approach for the longest period of time.

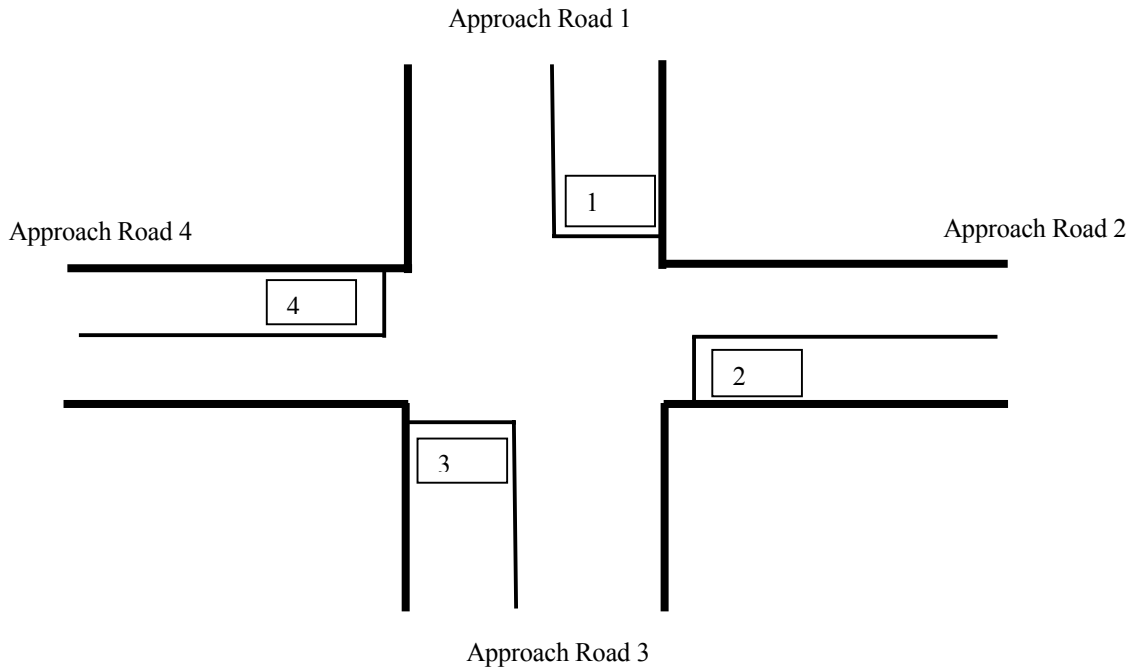


Figure 3.2: Example 4WS scenario, illustrating rule of right of way granting.

An algorithm is presented in the next chapter which will be performed by vehicles to determine which vehicle has right of way or which vehicle is entitled to right of way. This algorithm is based on the 4WS shared data structure also detailed in the next chapter.

It must be noted that the ordering of right of way granting to vehicles is relatively unimportant to this dissertation. This is due to the fact that deciding which vehicle has right of way at any one time is based on an *interpretation* of the vehicles' shared data structure (detailed later). Future research could involve revising this interpretation of the shared data structure to ensure maximum efficiency or fairness for all concerned vehicles. Of primary concern to this dissertation is the *correctness* and *maintenance* of the shared data structure, not the interpretation thereof. We are investigating if current group communication services can accurately maintain the shared data structure.

3.3 4WS Automated Vehicle Model

Automation is an important focus of current research in transportation systems. Research indicates that vehicle automation will lead to improved transportation systems through

increased safety, decreased traffic congestion and improved environmental pollution [49, 50, 51].

We are concerned with a specific automated vehicle model which will operate in a 4WS implementation. A 4WS vehicle is expected to be intelligent and sentient, in that it must be aware of its surrounding environment and be able to react reasonably to changes in the environment. Such vehicle sentience, will be provided by a diverse range of onboard sensors and decentralised vehicle communication. At a minimum, for implementation of a 4WS application, the required features of a 4WS vehicle will be provided by: global position awareness using GPS, digital road maps, vehicle obstacle detection and wireless communication ability.

A 4WS vehicle will be more or less aware of its accurate global location. This will be achieved by a system integrating accurate GPS and digital road maps, with the effect that vehicles can determine what road they are on, which direction they are travelling, what junction they are approaching etc.

Using obstacle detection technology, 4WS vehicles will be able to detect the presence of an obstacle in front of the vehicle. Such vehicle obstacle detection will be accurate to a distance that exceeds the maximum braking distance of the vehicle. This maximum braking distance is in turn dependent on the maximum allowable speed of the vehicle. Although, vehicles will be able to detect the presence of an obstacle, they will be unable to identify the obstacle. For example, when an obstacle is detected, the vehicle will not be able to determine if the obstacle is another vehicle or a pedestrian etc.

Another feature of 4WS vehicle, is its ability to wirelessly communicate with other 4WS vehicles. This will be accomplished using wireless 802.11 transmitters and receivers. This hardware will enable vehicles to wirelessly communicate in a decentralised manner. This wireless communication will enable vehicles to inform each other about their common environment. In conjunction with GPS and digital roadmaps, vehicles can enrich messages with location information.

An intelligent vehicle may also have other sensors or features, not mentioned above. Other optional sensors may be incorporated into an intelligent 4WS vehicle, but are not

directly required for a 4WS implementation. It must also be noted that this dissertation is only concerned the problem posed at a 4WS: which vehicle has right of way to cross the junction. Other protocols may be responsible for actually driving and manoeuvring the vehicle in the vicinity of the junction.

3.4 Application-Level Traffic Scenarios

In order to develop a suitable shared data structure which can be accurately and unambiguously interpreted to determine which vehicle at the 4WS junction should seize right of way, we must first look at all possible application-level scenarios at the junction. Eleven general scenarios were identified which classify all possible traffic scenarios at the 4WS. These scenarios were used to identify group communication primitives and features of a suitable share data structure, both of which are detailed in chapter four. The identified application level scenarios are detailed now:

1. An empty junction:
 - a. In the initial case, the 4WS junction and its surrounding area is empty of vehicles interested in gaining right of way at the junction (figure 3.3).

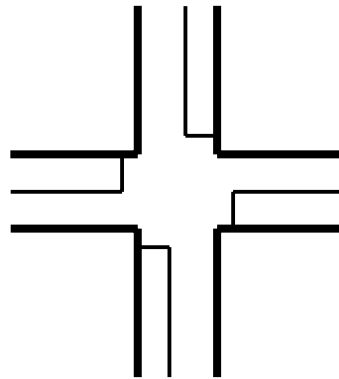


Figure 3.3: An empty junction.

2. Vehicle(s) approach an empty junction:
 - a. A single vehicle approaches an empty junction (figure 3.4a);
 - b. More than one vehicle approach an empty junction “simultaneously” (figure 3.4b).

Note, that by “simultaneously”, we mean that the vehicles enter the vicinity of the 4WS junction at approximately the same time.

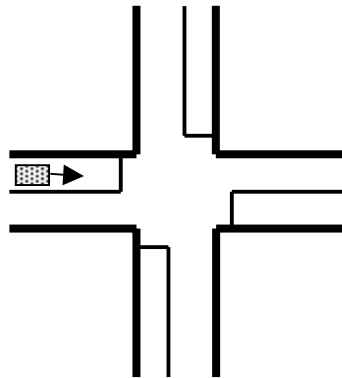


Figure 3.4a: A single vehicle approaching an empty junction.

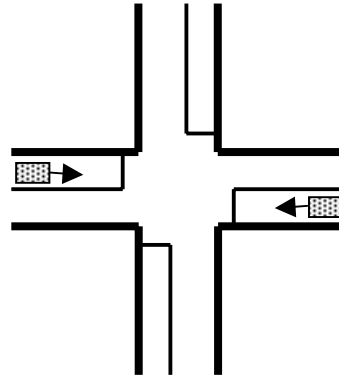


Figure 3.4b: Multiple vehicles approaching an empty junction “simultaneously”.

3. A vehicle approaches a non-empty junction:

- a. A vehicle approaches a non-empty junction along an empty approach road (figure 3.5a);
- b. A vehicle approaches a non-empty junction along a non-empty approach road (figure 3.5b).

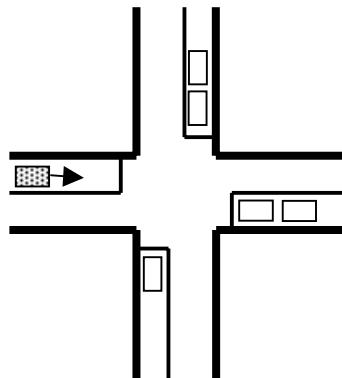


Figure 3.5a: Vehicle approaches a non-empty junc. along an empty approach road.

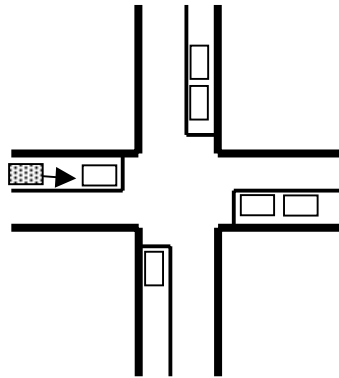


Figure 3.5b: Vehicle approaches a non-empty junc. along a non-empty approach road.

4. A vehicle seizes right of way:

- a. Based on an interpretation of the shared data structure, a vehicle determines that it should have right of way at the junction. It then seizes the right of way (figure 3.6) with the intention of crossing the junction.

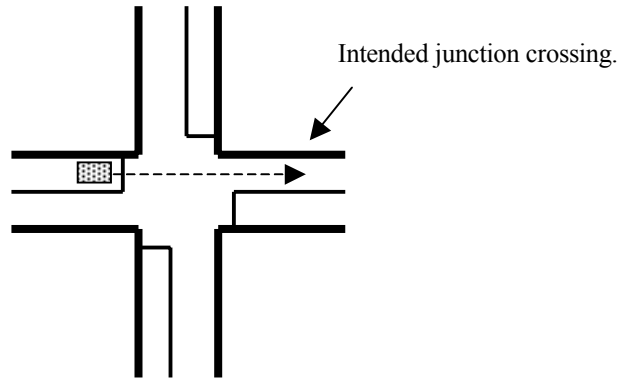


Figure 3.6: Vehicle seizes right of way (with the *intention* of crossing the junction).

5. A vehicle leaves junction after seizing right of way:
 - a. A vehicle has seized right of way then crossed the junction leaving the junction subsequently empty of interested vehicles (figure 3.7a);
 - b. A vehicle has seized right of way then crossed the junction leaving the junction subsequently non-empty of interested vehicles (figure 3.7b).

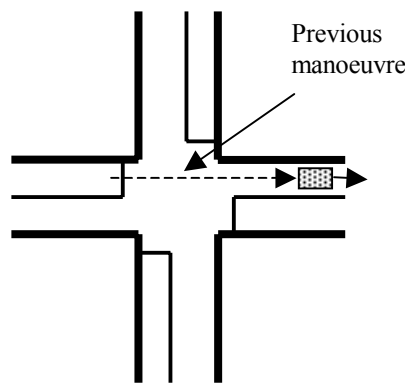


Figure 3.7a: A vehicle seized right of way and left the junction subsequently empty.

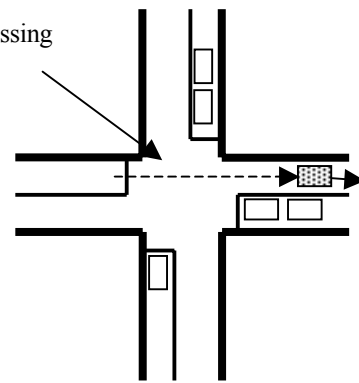


Figure 3.7b: A vehicle seized right of way and left the junction subsequently non-empty.

6. A vehicle leaves 4WS queue before seizing right of way (e.g. vehicle “pulls in” or performs a u-turn):
 - a. A vehicle leaves 4WS queue leaving the junction subsequently empty (figure 3.8a);

- b. A vehicle leaves 4WS queue leaving junction subsequently non-empty (figure 3.8b).

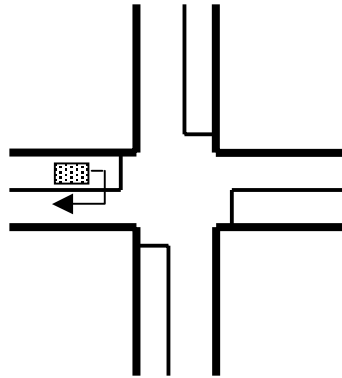


Figure 3.8a: A vehicle, before seizing RoW, leaves the junction subsequently empty.

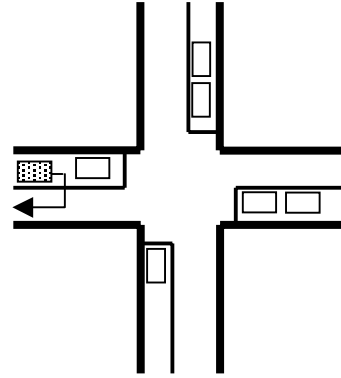


Figure 3.8b: A vehicle, before seizing RoW, leaves the junction subsequently non-empty.

7. A vehicle stalls:

- a. A vehicle stalls at the front of an approach road (figure 3.9a);
- b. A vehicle stalls along an approach road, but not the front of the approach road (figure 3.9b);
- c. A vehicle stalls on the junction (figure 3.9c).

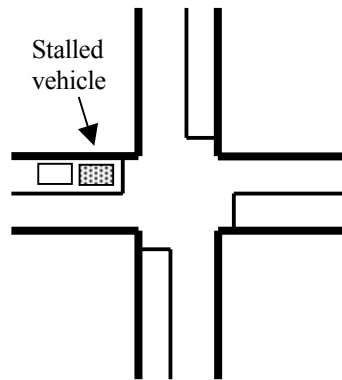


Figure 3.9a: A vehicle stalls at the front of an approach road.

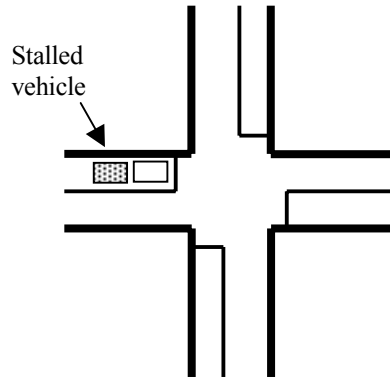


Figure 3.9b: A vehicle stalls on an approach road, but not the front.

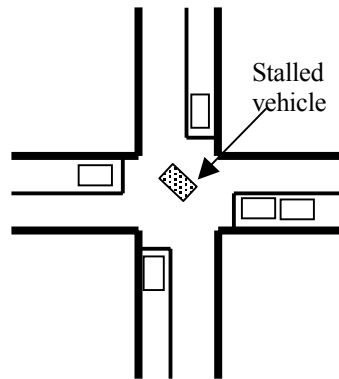


Figure 3.9c: A vehicle stalls on the junction.

Definition: A “local vehicle” is a vehicle that is in the proximity of a 4WS junction but that is not interested in crossing the junction. For example: a vehicle that has already crossed the junction and is driving away from the junction, or a vehicle that is travelling on an approach road but that intends pulling in before the junction.

8. A “local” vehicle obstruction:

- a. A local vehicle causes an obstruction at the front of an approach road (figure 10.a);
- b. A local vehicle causes an obstruction along an approach road, but not the front of the approach road (figure 10.b);
- c. A local vehicle causes an obstruction on the junction (figure 10.c).

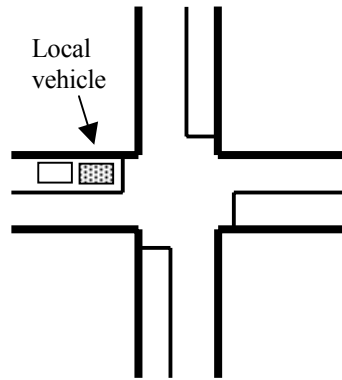


Figure 3.10a: A local vehicle obstruction at the front of an approach road.

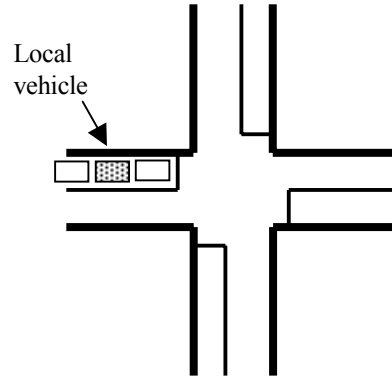


Figure 3.10b: A local vehicle obstruction on an approach road, but not at the front.

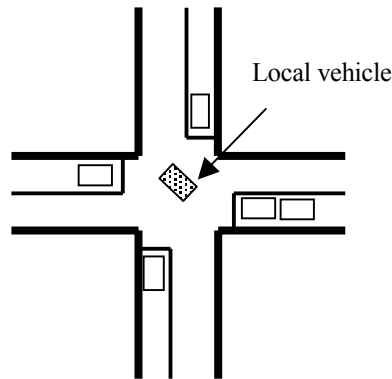


Figure 3.10c: A local vehicle obstruction on the junction.

9. A vehicle relinquishes right of way before leaving the junction:
 - a. A vehicle relinquishes right of way before entering on to the junction (figure 3.11);

Note, a vehicle may not relinquish right of way while on the junction. For example, if a vehicle stalled on the junction, it should not relinquish right of way, because to do so would imply that another vehicle should cross the road while the stalled vehicle is still present on the junction. This could lead to possible collisions.

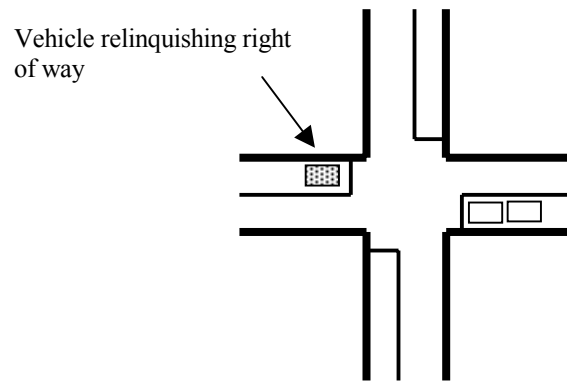


Figure 3.11: Vehicle relinquishes right of way before entering on to the junction.

10. A vehicle reports a blockage:

- a. A vehicle reports a blockage at the front of an approach road (figure 3.12a);
- b. A vehicle reports a blockage along an approach road, but not the front of the approach road (figure 3.12b);
- c. A vehicle reports a blockage on the junction (figure 3.12c).

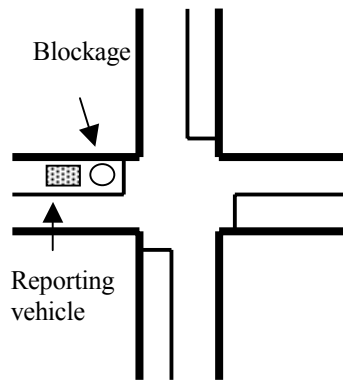


Figure 3.12a: A vehicle reports a blockage at the front of an approach road..

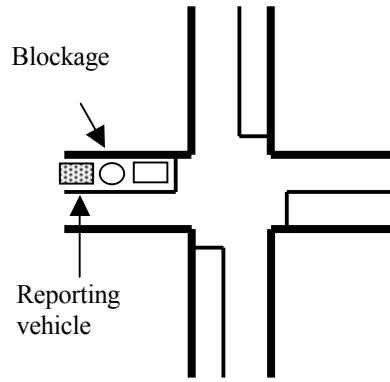


Figure 3.12b: A vehicle reports a blockage along an approach road, but not the front.

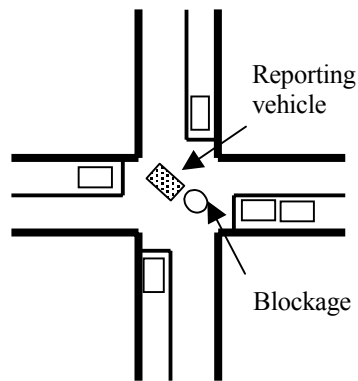


Figure 3.12c: A vehicle reports a blockage on the junction.

11. Vehicle reports removal of a blockage:

- a. A vehicle reports the removal of a blockage previously on the junction;
- b. A vehicle reports the removal of a blockage previously on an approach road.

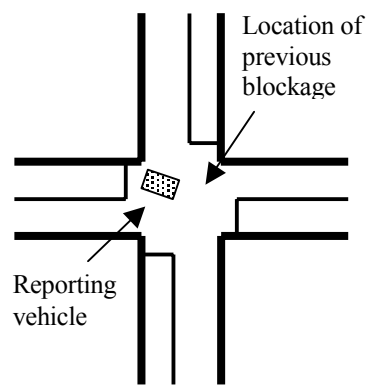


Figure 3.13a: Vehicle reports a removal of a blockage on the junction.

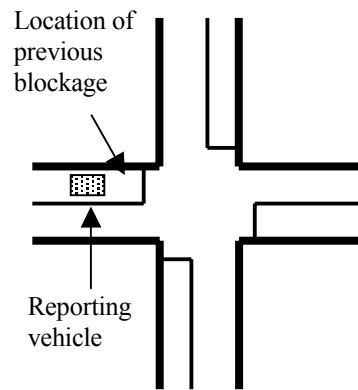


Figure 3.13b: Vehicle reports a removal blockage of a blockage along an approach road.

This chapter introduced the background behind the 4 Way Stop problem. The rule for granting right of way to vehicles was described and application-level 4WS traffic scenarios were detailed. The next chapter builds on this background and makes steps towards implementing a 4WS application using group communication. Group communication primitives and a suitable 4WS shared data structure are discussed.

Chapter 4: Mapping 4 Way Stop Problem to Group Communication

In order to implement a 4WS protocol to coordinate vehicles, we must identify group communication primitives based on the application-level scenarios detailed in the previous chapter. Also, we must understand membership of the 4WS group. Finally, we must develop a shared data structure to represent the traffic scenario, and identify the requirements on the shared data structure.

4.1 Membership Interest

Following on from the definition of *proximity groups* in [42], which states that membership of groups in mobile systems should depend on both location and on functional aspects, only certain vehicles within the 4WS junction's proximity will be involved in determining who has right of way. This is necessitated by the fact that some vehicles in the vicinity of the junction may not be interested in joining the group: for example vehicles intent on "pulling in" before the junction or vehicles performing a u-turn. Also, vehicles driving away from the junction do not need to involve themselves in group communication, however, they may have previously been involved. A vehicle's presence in proximity to the junction is not sufficient for membership in the proximity group: a vehicle must explicitly specify its interest in joining the group and hence its interest in eventually gaining right of way at the junction.

4.2 Group Communication Primitives

From the application level traffic scenarios listed previously, we have identified the following 4WS group communication primitives which if successfully implemented will enable the 4WS shared data structure to be accurately maintained through mutual assistance of member vehicles. The purpose of each primitive is detailed in this section,

however, later in the chapter, the effects of each of these primitives on the shared data structure will be discussed.

1. Create:

A vehicle creates a proximity group when it approaches a previously empty 4WS junction. This will involve, creating and initialising the 4WS shared data structure.

2. Join:

A vehicle issues this join primitive when it wishes to join the proximity group around a 4WS junction with the intention of eventually seizing right of way at the junction.

3. Expected leave:

This primitive is issued by a vehicle when it successfully crossed the 4WS junction and wishes to leave the proximity group.

4. Unexpected leave:

This primitive is issued by a vehicle when it wishes to leave the proximity group before seizing right of way at the junction. This primitive is used when a vehicle performs a u-turn, for example, after deciding it is no longer interested in gaining right of way at the junction.

5. Seize right of way:

After interpreting the shared data structure and determining that it should have right of way, this primitive is issued by a vehicle before attempting to cross the junction.

6. Relinquish right of way:

A vehicle may relinquish right of way using this primitive. If a vehicle is to issue this primitive, it must also have previously reported a blockage on an approach road (primitives 8).

7. Report junction blocked:

A vehicle can report the junction as blocked using this primitive. This may be done because of an unexpected obstacle detected on the junction, or due to the vehicle itself stalling on the junction.

8. Report approach blocked:

A vehicle can report an approach road as blocked using this primitive. This may be done because of an unexpected obstacle detected on the approach, or due to the vehicle itself stalling on the approach.

9. Report removal of blockage on junction:

This primitive is used by a vehicle to report the removal of a blockage on the junction.

10. Report removal of blockage on an approach road:

This primitive is used by a vehicle to report the removal of a blockage on an approach road.

4.2.1 Ordering of Group Communication Primitives

The above group communication primitives should be subject to certain ordering constraints. Primitive ordering is classified into local ordering and global ordering. Local ordering means that primitives must be issued by individual vehicles in specific orders, for example a vehicle can only issue an *unexpected leave* primitive after issuing a *join* primitive. Global ordering enforces an ordering on primitives being issued among all member vehicles, for example one vehicle must issue the *create* primitive before another vehicle can issue the *join* primitive.

Local ordering of primitives:

- *join* before *seize right of way*:

A vehicle must have joined the proximity group before it can possibly seize right of way at the junction.

- *seize right of way before expected leave:*
A vehicle must have seized right of way and successfully crossed the junction before it can leave the group by issuing an expected leave primitive.
- *join before unexpected leave:*
A vehicle must have joined the proximity group before it can leave the group by issuing an unexpected leave primitive.
- *Seize right of way before report approach blocked:*
A vehicle must have seized right of way, and still be in possession of the right of way, before it can report an approach as blocked. This has the effect of ensuring that only one car can report a blockage along an approach at any one time, thus limiting shared data structure updates.
- *Seize right of way before report junction blocked:*
A vehicle must have seized right of way, and still be in possession of the right of way, before it can report the junction as blocked. This has the effect of ensuring that only one car can report a blockage on the junction at any one time, thus limiting shared data structure updates.
- *report approach blocked before relinquish right of way:*
A vehicle can only relinquish right of way if it has reported an approach as blocked. This is because, if the vehicle cannot successfully cross the junction after seizing right of way then there must exist a blockage of some form (foreign object, other vehicle or the vehicle itself stalled) at the front of the approach, which is required to be reported to other vehicles.

Global ordering of primitives:

- *Create before join:*
One vehicle must have established a proximity group using the create primitive before another vehicle can join the group.

- *report approach blocked before report removal of blockage on approach road:*
An approach must have been reported as blocked by one vehicle before another vehicle can report the approach as unblocked. The vehicle reporting the approach as unblocked may have been the same vehicle that reported the approach as blocked, but not necessarily so.
- *report junction blocked before report removal of blockage on junction:*
The junction must have been reported as blocked by one vehicle before another vehicle can report the junction as unblocked. The vehicle reporting the junction as unblocked may have been the same vehicle that reported the junction as blocked, but not necessarily so.

The above ordering constraints can be nested to give full ordering constraints for any particular primitive. For example, when a vehicle reports the removal of a blockage on an approach, the following is a possible ordering of primitives:

Create before join (global); join before seize right of way (local); seize right of way before report approach blocked (local); report approach blocked before report removal of blockage on approach (global).

4.3 Shared Data Structure

In order for the 4WS protocol to be successfully implemented, it is necessary that a suitable data structure be successfully shared among, and maintained by, all proximity group member vehicles. If the data structure is successfully distributed amongst all interested group members then, all members can independently determine which vehicle is next to be granted right of way at the 4WS junction by interpreting the shared data structure. A level of sentience is achieved when vehicles independently determine which vehicle has right of way. This is due to the fact that vehicles identified changes in their environment (4WS), through group communication, obstacle detection, GPS and digital roadmaps, and acted upon these changes to control crossing the junction.

The proposed shared data structure is composed of three elements:

1. Membership Queue:

This element lists the member vehicles according to the approach road (1 to 4) on which they queue and also orders the vehicles according to their actual physical ordering upon the approach road.

2. Right of Way History:

This element is an historical record of the ordering of right of ways granted to each of the four approach roads.

3. Blockage Status:

This element contains a Boolean status flag for each of the four approach roads. These Booleans represent each of the approaches' blockage status. For example, if an approach has been reported as blocked, then its Boolean will be set to true, otherwise the Boolean is set to false. There is also a Boolean to represent the blockage status of the entire junction. If the junction is reported as blocked then the Boolean is set to true, otherwise it is set to false.

These data structure elements are best explained by examples.

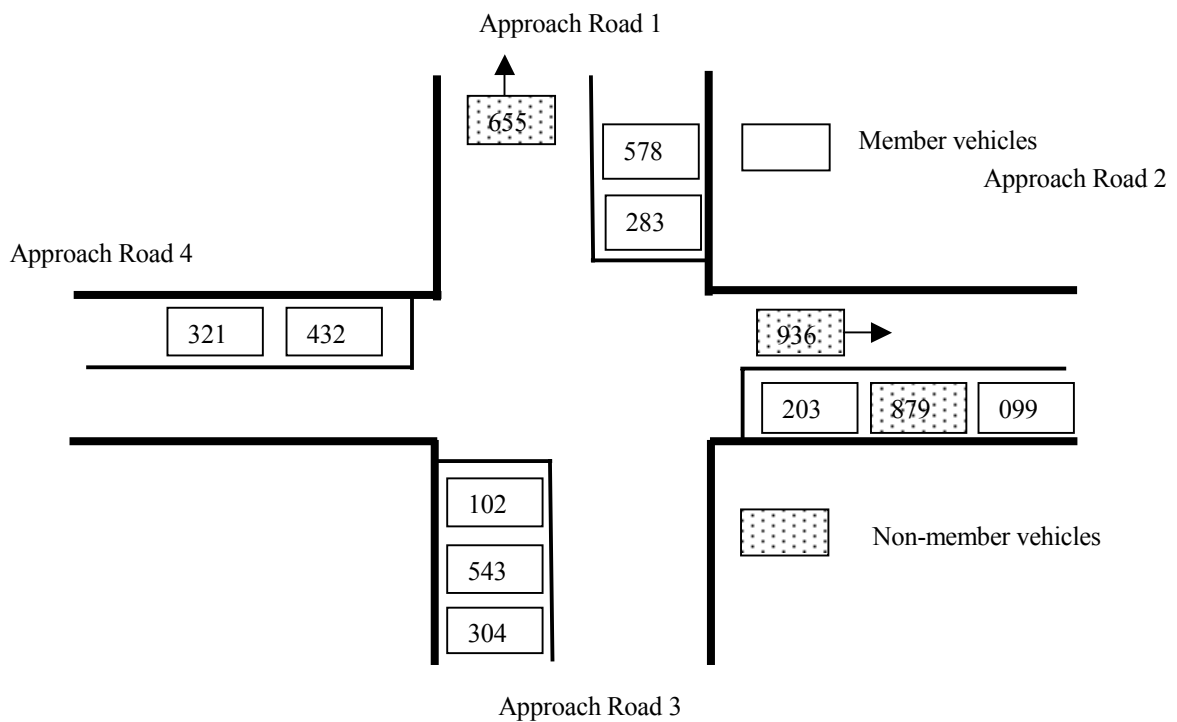


Figure 4.1: Example Traffic Scenario.

In figure 4.1, there are twelve vehicles in the vicinity of the junction: ten vehicles queuing on the four approach lanes and two vehicles driving away from the junction. Although, vehicles 879, 936 and 655 are within the geographical area of the junction, they are not members of the 4WS group. Vehicle 936 and 655 are driving away from the junction and hence are not interested in receiving right of way to cross the junction. Vehicle 879 is queuing along approach road, yet has not joined the group. This “local vehicle” may be about to “pull in” before the junction or might be about to perform a u-turn or may even be broken down. Perhaps, the vehicle intends on joining the group, but has not yet. This, however, is only speculation: the fact remains that the vehicle is not a group member. Taking all this into account, the *Membership Queue* data structure element for this scenario will look like that shown in figure 4.2:

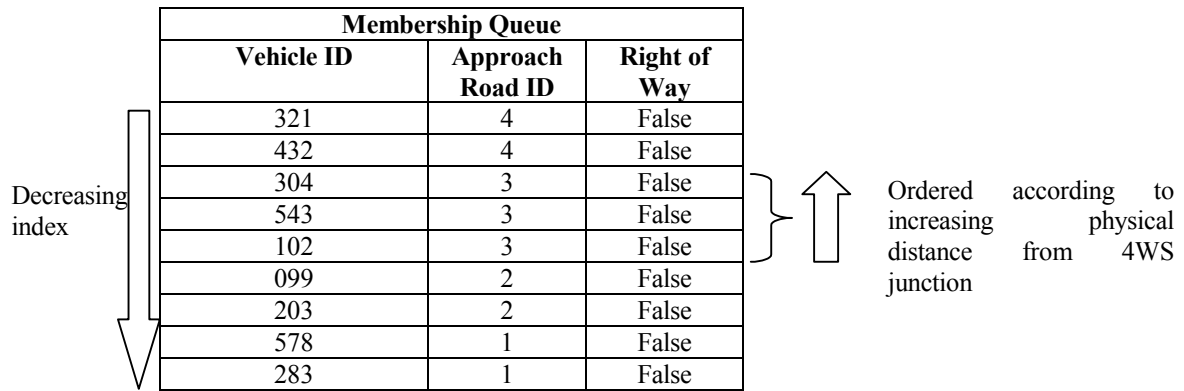


Figure 4.2: Element 1: “Membership Queue”.

Please note, that no global ordering exists throughout this data structure element. For example vehicle 283 being indexed lower than vehicle 321 in the data structure does not hold any significance. However, there does exist an ordering within the data structure element with regards to each of the approach road queues. Take the approach road queue consisting of vehicles on approach 3. This queue contains vehicles 102, 543 and 304. The indexing of vehicles within this approach road queue is significant: the significance is that the three vehicles are ordered within the approach road queue according to their actual physical ordering on the approach road (i.e. their physical distance from the 4WS junction). That is vehicle 102 is physically in front of vehicle 543, which in turn is in front of vehicle 304. This will be a very important requirement on the 4WS application: group membership joins must be added to the “Membership Queue” in a manner respecting the joining vehicle’s actual physical ordering on the approach road.

The second data structure element is the *Right of Way History*. This historical data structure element consists of four fields, one field for each of the four approach roads. The fields represent the order in which vehicles on each of the approach roads last received right of way at the junction and hence it represents the future right of way priority of an approach. Consider the *Right of Way History* in figure 4.3: this log shows that the last approach to receive right of way at the junction was approach 4 and the approach longest waiting for right of way is approach 3. Hence approach 3 should be the next approach to receive right of way. This is of course assuming that there are vehicles queuing on approach 3, which is the case on this example scenario. If in a scenario some approach

roads were empty of queuing vehicles, then right of way is given to the highest prioritised non-empty approach road.

Right of Way History	
Approach Road ID	Approach Priority
1	2
2	3
3	1
4	4

Figure 4.3: Element 2: “Right of Way History”.

The third and final data structure element is the *Blockage Status*. This element is comprised of five Booleans, one for each of the four approaches and a fifth for the junction itself. The fields represent the current status of each of the approaches and the junction with regard any possible blockages. An approach is considered blocked if and only if *all vehicles*, interested in gaining right of way at the 4WS junction, along the approach in question cannot make progress towards the 4WS due to a blockage. Example blockages include: a fallen tree, a stalled car, road works, pedestrians, a local vehicle obstruction etc. Consider the scenarios shown in figure 4.4 and figure 4.6. As can be seen in figure 4.4 a local vehicle has blocked approach 3 such that *no* member cars on the approach can make progress. In this case the *blockage status* should look like figure 4.5, assuming approaches 1, 3 and 4 (unshown in figure 4.4) are unblocked. As can be seen, the entry corresponding to approach road 3 indicates that the approach is blocked and no vehicles along approach 3 can make progress. The scenario illustrated in figure 4.6 is different however: not all the vehicles along approach road 3 are blocked, only vehicle 035. Again, according to the above definition of a blocked approach, *all* vehicles must be unable to make progress. Hence, the blockage status data structure element corresponding to figure 4.6 will show approach road 3 to be unblocked, although, after vehicle 903 has seized right of way, approach road 3 will be reported as blocked assuming the local vehicle is still causing an obstruction.

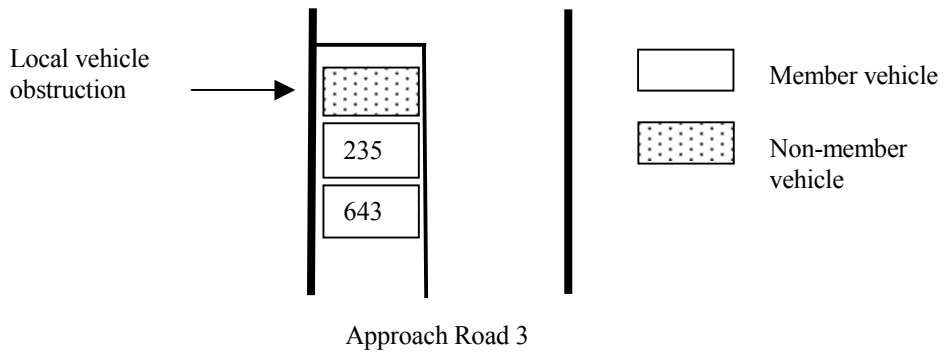


Figure 4.4: Local vehicle blocking all vehicles on approach road 3.

Blockage Status	
Approach Road ID / Junction	Blockage Status (Blocked=true)
Junction	False
1	False
2	False
3	<i>True</i>
4	False

Figure 4.5: Element 3: “Blockage Status”.

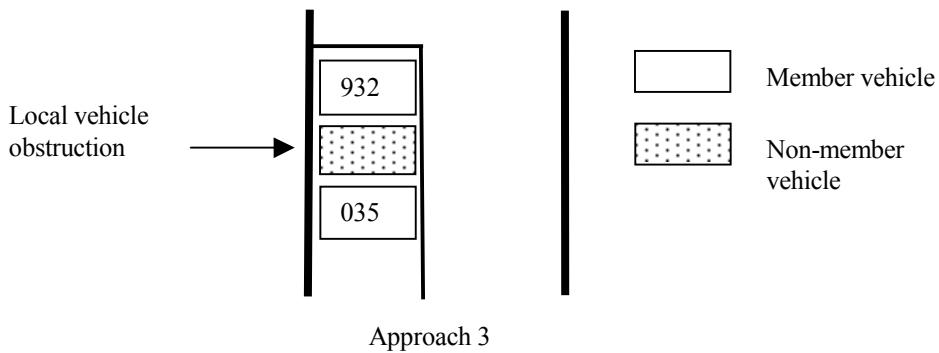


Figure 4.6: Local vehicle blocking subset of vehicles on approach 3.

The blockage status data structure element also contains a field to represent the blockage status of the junction itself. The junction should be reported as blocked, by a vehicle in possession of right of way, if any unexpected obstruction exists on the junction. Reporting of a junction as blocked will have the effect of causing the 4WS application of also

blocking. No vehicles should attempt to cross the junction if the junction is blocked. All vehicles must simply wait for the obstruction to be reported as cleared before progress can be made.

4.3.1 Maintenance of Shared Data Structure using Primitives

To show how the shared data structure is maintained using the 4WS group communication primitives this section explained shared data structure updates for each primitive.

Create:

The group-creating vehicle will create a data structure and initialise it. The membership queue data structure element shall contain only a single entry containing the creating vehicles id and its approach road number. The right of way history log will be initialised to prioritise the approach on which the creating vehicle approaches the junction. The other approaches' priorities will be set in a clockwise manner starting from the creating vehicle's approach. The blockage status data structure element will also be initialised to assume that no blockages exist on any of the four approach roads or on the junction. For example, assuming a group-creating vehicle of id 915, say, approach a 4WS junction along approach number 1, the newly created data structure elements will look like that shown in figure 4.7.

Membership Queue		
Vehicle ID	Approach Road ID	Right of Way
915	1	False

Right of Way History	
Approach Road ID	Approach Priority
1	1
2	2
3	3
4	4

Blockage Status	
Approach Road ID / Junction	Blockage Status (Blocked=true)
Junction	False
1	False
2	False
3	False
4	False

Figure 4.7: Example data structure elements after group is created.

Join:

When a vehicle joins an established proximity group, the only required change to the data structure is to the membership queue. The joining vehicle's id and the number of the approach road on which the vehicle is approaching the junction are added to the data structure element. If a vehicle joins the group on an approach road which was previously empty of member vehicles, then the joining vehicle is simply added to the membership queue with no ordering constraints. However, if the approach road has other member vehicles queuing for right of way then an ordering constraint does exist on the membership queue: the joining vehicle's associated membership queue entry must be added to the data structure element with an index higher than all membership queue entries of vehicles physically ordered in front of the joining vehicle and with an index lower than all membership queue entries of vehicles physically order behind the joining vehicle. For example, say vehicle of id 407 joins a group while approaching a junction on approach road 2. Also, assume that on approach road 2, two vehicles of id 200 and 943 are already group members with vehicle 200 physically in front the joining vehicle and vehicle 943 physically behind the joining vehicle. This scenario, shown in figure 4.8, could occur if vehicle 943 somehow managed to join the group before 407. Vehicle 407 may for example have pulled out in front of vehicle 943.

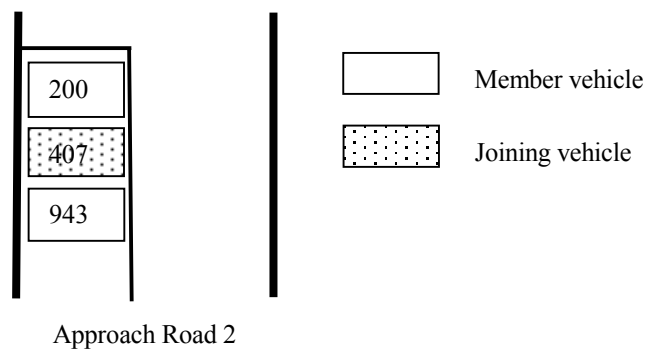


Figure 4.8: Example “join” traffic scenario.

Such a join scenario will be required to update the membership queue as shown in figure 4.9, assuming that no other member vehicles have joined the group.

Membership Queue		
Vehicle ID	Approach Road ID	Right of Way
943	1	False
407	1	False
200	1	False

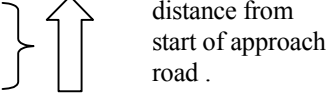


Figure 4.9: Membership queue update after join.

Expected leave:

An expected leave has the result of removing a member vehicle from the data structure. The member vehicle will be in possession of right of way when the removal occurs. An example membership queue update is shown in figure 4.10 when vehicle id 768 leaves the group after crossing the junction.

Membership Queue (before)			Membership Queue(after)		
Vehicle ID	Approach Road ID	Right of Way	Vehicle ID	Approach Road ID	Right of Way
303	4	False	303	4	False
768	2	True	285	1	False
285	1	False			

Figure 4.10: Before and after membership queue for an “expected leave”.

An expected leave also has an effect on the right of way history data structure element. The approach which is now releasing right of way (approach road 2), in the form of vehicle 768’s expected leave, will be the least prioritised approach (new priority of 4). All other approach roads’ priorities are decreased by one. Figure 4.11 shows that approach road 1 is the highest prioritised approach after the expected leave.

Right of Way History (before)		Right of Way History (after)	
Approach Road ID	Approach Priority	Approach Road ID	Approach Priority
1	2	1	1
2	1	2	4
3	3	3	2
4	4	4	3

Figure 4.11: Before and after right of way history for an “expected leave”.

Unexpected leave:

An expected leave has the same effect on the membership queue data structure element as an expected leave, above, except that the leaving vehicle was not previously in possession of right of way before leaving the group. Although both “leave” primitives could be implemented as a single primitive, with the “expected” and “unexpected” semantics determined locally by an interpretation of the data structure change, it was decided to allow two specific leave primitives for the sake of enforcing ordering constraints on when leave events can occur.

An unexpected leave has no effect on any data structure elements other than the membership queue.

Seize right of way:

The seize right of way primitive is issued by a vehicle when it has determined that it is due to receive right of way. A vehicle issuing a seize right of way primitive changes its membership queue entry to indicate that it now has right of way as shown in figure 4.12.

Membership Queue (before)			Membership Queue (after)		
Vehicle ID	Approach Road ID	Right of Way	Vehicle ID	Approach Road ID	Right of Way
285	1	False	285	1	True

Figure 4.12: Before and after membership queue for “seize right of way”.

Relinquish right of way:

A vehicle issues a relinquish right of way primitive if it cannot cross the junction for some reason e.g. vehicle stalled or obstruction. Such a primitive will be follow a blocking report approach primitive (primitive 8). The effect of the relinquish right of way primitive, as shown in figure 4.13, is that the vehicle’s membership queue entry is updated to show that the vehicle no longer is in possession of the right of way.

Membership Queue (before)			Membership Queue (after)		
Vehicle ID	Approach Road ID	Right of Way	Vehicle ID	Approach Road ID	Right of Way
285	1	True	285	1	False

Figure 4.13: Before and after membership queue for “relinquish right of way”.

Report junction blocked:

A vehicle, in possession of the right of way, can report the junction as blocked if it detects an obstruction on the junction preventing the vehicle from crossing the junction or if the vehicle itself has stalled on the junction. The associated data structure update, affecting the blockage status element, changes the junction status to indicate the blockage. This in turn will have the effect of stopping all vehicles from proceeding across the junction. An example update is shown in figure 4.14.

Blockage Status (before)		Blockage Status (after)	
Approach Road ID / Junction	Blockage Status (Blocked=true)	Approach Road ID / Junction	Blockage Status (Blocked=true)
Junction	False	Junction	True
1	False	1	False
2	False	2	False
3	False	3	False
4	False	4	False

Figure: 4.14: Before and after blockage status for “report junction blocked”.

Report approach blocked:

A vehicle, in possession of the right of way, can report the approach, on which it is located, as blocked if it detects an obstruction at the front of the approach preventing the vehicle from crossing the junction or if the vehicle itself has stalled. The associated data structure update, affecting the blockage status element, changes the affected approach status to indicate the blockage. This in turn will have the effect of stopping all vehicles on that approach from gaining right of way. An example update for a blockage detected on approach 2 is shown in figure 4.15.

Blockage Status (before)		Blockage Status (after)	
Approach Road ID / Junction	Blockage Status (Blocked=true)	Approach Road ID / Junction	Blockage Status (Blocked=true)
Junction	False	Junction	False
1	False	1	False
2	False	2	True
3	False	3	False
4	False	4	False

Figure: 4.15: Before and after blockage status for “report approach blocked”.

Report removal of blockage on junction:

A vehicle can report the removal of a blockage on the junction after an obstruction was previously reported on the junction by a vehicle. This occurs when a vehicle no longer detects a blockage on the junction. The associated data structure update, affecting the blockage status element, changes the junction status to indicate the removal of the blockage. Vehicles will be able to resume normal operation following the issuance of this primitive. An example update for is shown in figure 4.16.

Blockage Status (before)		Blockage Status (after)	
Approach Road ID / Junction	Blockage Status (Blocked=true)	Approach Road ID / Junction	Blockage Status (Blocked=true)
Junction	True	Junction	False
1	False	1	False
2	False	2	False
3	False	3	False
4	False	4	False

Figure: 4.16: Before and after blockage status for removal of blockage on junc.

Report removal of blockage on an approach road:

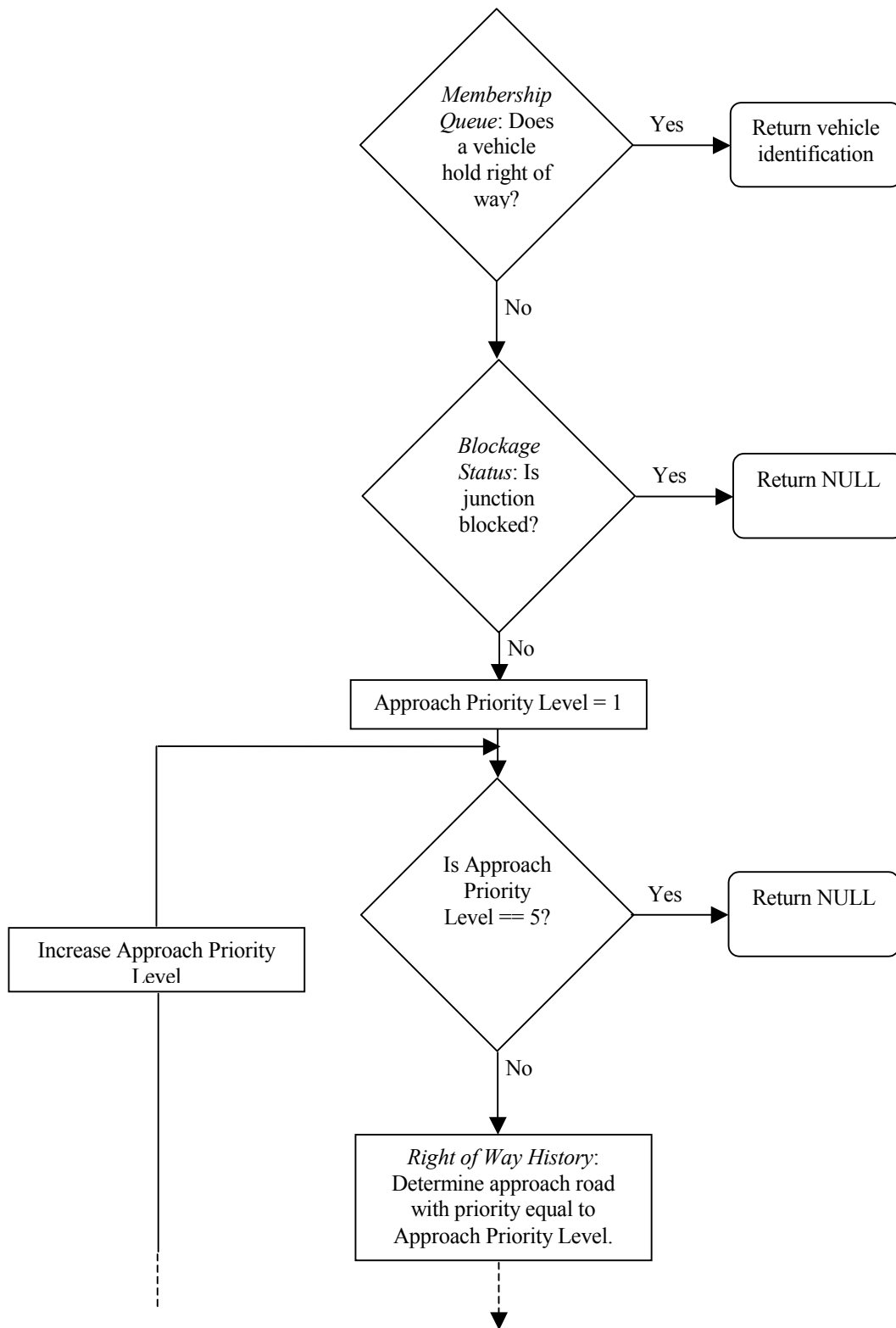
A vehicle can report the removal of a blockage on an approach road after an obstruction was reported on the approach by a vehicle. This occurs when a vehicle no longer detects the presence a blockage on the approach. The associated data structure update, affecting the blockage status element, changes the affected approach road status to indicate the removal of the blockage. Vehicles on the affected approach will be able to resume normal operation following the issuance of this primitive, assuming the junction is unblocked. An example update for a is shown in figure 4.17.

Blockage Status (before)		Blockage Status (after)	
Approach Road ID / Junction	Blockage Status (Blocked=true)	Approach Road ID / Junction	Blockage Status (Blocked=true)
Junction	False	Junction	False
1	True	1	False
2	False	2	False
3	False	3	False
4	False	4	False

Figure 4.17: Before and after blockage status for removal of blockage on approach.

4.3.2 Right of Way Algorithm

The following algorithm is used to determine which vehicle currently has right of way or which vehicle is entitled to seize right of way at the 4WS junction. The algorithm is based on the 4WS shared data structure. If the shared data structure is correctly maintained then this algorithm can be used by each vehicle to locally determine if it is seize right of way at the junction.



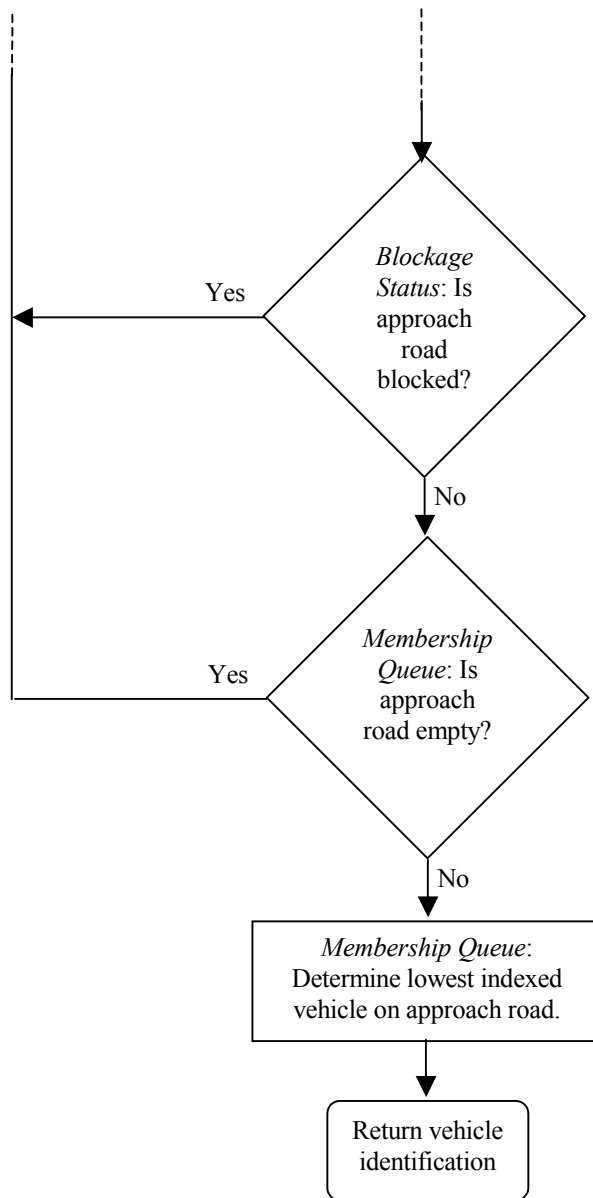


Figure 4.18: Right of Way algorithm.

If the ten 4WS group communication primitives are successfully implemented to maintain the correctness of the shared data structure as described in this chapter, then the a 4WS group communication protocol could be developed to reliably determine vehicle right of way at a 4WS junction using the algorithm detailed above. The next chapter shall list the requirements on a group communication service to successfully implement a replication service to maintain the global consistency of the shared data structure. The chapter shall detail the requirements for both a primary component implementation and a partitionable membership implementation.

Chapter 5: Group Communication Requirements

This chapter examines the requirements on group communication to successfully support the 4WS inter-vehicle coordination application. Group communication will be required to maintain the consistency of the replicated 4WS shared data structure. We provide specific group communication requirements for two optional 4WS group communication-based implementations: primary component membership and partitionable membership.

Related Work

The group communication requirements considered in this chapter are based on those defined by Chockler, Keidar and Vitenberg in their exhaustive survey of over thirty published group communication specifications [23]. Chockler et al. provide a comprehensive set of clear and rigorous group communication requirements. These requirements can be combined to provide the required guarantees of most existing group communication systems.

The paper provides a unifying framework for analysis, comparison and classification of group communication protocols. The authors' work was fuelled by the fact that many GCSs use different terminologies to express their respective requirements. Also, some expressed requirements were found to be ambiguous, thus increasing the difficulty in analysing and comparing GCSs.

Primary component and partitionable membership services

A group membership service may be either primary component or partitionable. In a primary component membership service, views installed by all the member processes in the system are totally ordered. However, in a partitionable membership service, views are only partially ordered. This implies that within a partitionable membership service,

multiple disjoint views may exist concurrently within the system. A GCS is partitionable if its membership service is partitionable; otherwise it is primary component. With regards the 4WS problem, a network partition could occur. In a primary component GCS, one network component identifies itself as primary. Vehicles in all other network components (non-primary) must rejoin the primary component group and be reissued the shared data structure from the primary component. However, in a partitionable implementation no network component identifies itself as primary: all network components are considered equals. Each component continues issuing dynamic, disjoint group views and, when a network merge occurs, vehicles within components perform state transfers to update their respective shared data structures with changes that occurred in other components during the partition.

Operating Environment: Fault Model and Assumptions

Any 4WS or other inter-vehicle coordination application implementation will be required to operate in an wireless network environment. Specifically, this dissertation investigates 4WS implementations operating in an infrastructure-free, ad hoc network wireless environment. As a result, an implementation (primary component or partitionable membership) would not require an expensive, time consuming deployment of wireless hardware infrastructure at all required junctions. All coordination would be managed on a peer-to-peer basis by the vehicles communicating amongst themselves and would therefore not be reliant on any form of upper hierarchy or infrastructure. The operating environment of an ad hoc network characteristics must be investigated to shape a fault model for a 4WS implementation.

The physical characteristics of an ad hoc network environment give rise to the fault model for the 4WS. These characteristics will, in turn, affect how each of the group communication properties described in this chapter are approached.

It is observed that in an ad hoc network any communication system is subject to *message omission* and *network partitions*:

Message omission is common in ad hoc networks primarily due to unpredictable node mobility and limited bandwidth availability.

With regard *network partitions*, it is noted that an ad hoc network may partition into a finite number of components. Processes in a component can receive multicasts from other processes within the component, but processes in two different components cannot communicate.

Some observations of mobile ad hoc networks are noted, but not addressed by this dissertation. We assume message corruption is not possible. Also it is assumed that Byzantine failures [53] do not occur, that is, processes do not behave in a malicious manner. Most group communication systems do not address Byzantine failures. Also, we assume network partitions will eventually heal.

In order for a 4WS application to be developed, any implementation must address the fault model laid out above, specifically, the issues of message omission and network partitioning.

Proposed System Architecture

The proposed system architecture is comprised of two layers (figure 5.1): 4WS application layer (application layer) and group communication service layer (group communication layer). The application layer integrates the necessary GPS, digital roadmaps and obstacle detection technologies as required. Also, the application layer, will have internal functionality to determine if a vehicle has right of way, based on the shared data structure interpretation using the algorithm detailed in the previous chapter. Embedded within the application layer is a shared data structure replication module. This replication module will maintain a local replica of the shared data structure. Updates to the local data structure replica are handled by the replication module based on the multicast semantics provided by the lower group communication layer. Finally, the group communication layer is primarily responsible for providing a totally ordered multicast to the upper layers based on simple send and receive operations to and from the underlying network. Requirements on the group communication layer are further detailed later in this chapter.

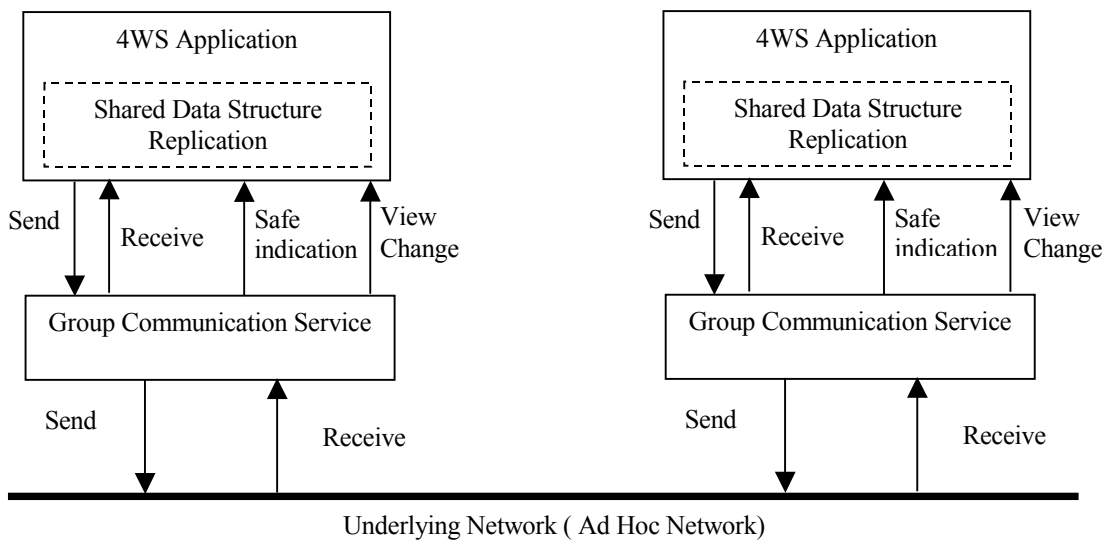


Figure 5.1: Proposed System Architecture.

Notation

This section summarises the notation used by Chockler et al. For more detailed references of the formal language used to describe the group communication properties, see [23]. The following are the basic sets used to formally define the group communication properties:

- P The set of processes within the system.
- M The set of messages sent by the application.
- V The set of views delivered to the application, where V is a pair containing $V.id$ and $V.elements$.
- VID The set of view identifiers, partially ordered by the $<$ operator.

Each action of the GCS is parameterised by a unique process p (an element of P) at which the action occurs. This means that one of the parameters of every GCS action must be the process which performs the action. The GCS interactions with the application are shown in figure 5.2 [23].

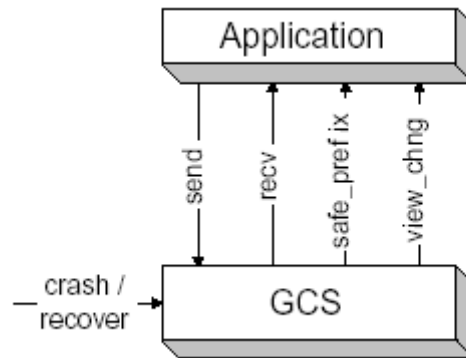


Figure 5.2: External actions of the GCS [23].

As can be seen, the application uses the GCS to both send and receive messages. The GCS also notifies the application of changes in the group view and indicates to the application when a message is safe. The external actions of the GCS are summarized in table 5.1.

$send(p, m)$	Process p sends message m .
$recv(p, m)$	Process p receives message m .
$view_chng(p, (id, members), T)$	Process p is informed of a group view change. The new group view is identified by $(id, members)$ where id is the group view id and $members$ are the member processes. T is the transitional set associated with the group view change.
$safe_prefix(p, m)$	Process p is indicated that message m is safe within the system.
$crash(p)$	Process p crashes.
$recover(p)$	Process p recovers after crashing.

Table 5.1: External actions of the GCS.

The group communication requirements are also formally stated using shorthand predicates. These predicates are summarised for reference in table 5.2.

$receives(p, m)$	Process p receives message m .
$receives_in(p, m, v)$	Process p receives message m in view v .
$sends(p, m)$	Process p sends message m .
$installs(p, v)$	Process p installs view v .
$installs_in(p, v, v')$	Process p installs view v in view v' .
$viewof(t_i)$	This returns the view in which event t_i occurred.
$receive_before(p, m, m')$	Process p receives message m before message m' .
$receive_before_in(p, m, m', v)$	Process p receives message m before message m' , both in view v .
$indicated_safe(p, m, v)$	Message m received in view v is indicated as safe at process p .
$stable(m, v)$	Message m is stable in view v .

Table 5.2: Shorthand Predicates.

With the formal notation used to express the group communication properties summarised, the next two sections shall describe the group communication requirements to successfully support a 4WS application. The first section is concerned with a 4WS implementation based on a primary component GCS. The subsequent section is concerned with an implementation based on a partitionable membership GCS.

Primary Component Membership Required Properties

As discussed previously, in a primary component membership GCS, views installed by all the member processes in the system are totally ordered. In a 4WS traffic scenario, a network partition could occur, meaning that certain vehicles in the vicinity of the 4WS junction cannot communicate with each other. This will mean that different network components have different views of their membership and hence will have different views of the traffic situation based on their respective, inconsistent shared data structures. In a primary component implementation one component will identify itself as primary (based on members that survive from the unpartitioned group to the partitioned group). This component will be the only component allowed to continue after a partition. Vehicles, not members of the primary component, will be expected to join the primary component after

the partition. The group communication requirements for a primary component 4WS implementation primarily involve maintaining a consistent shared data structure, within the primary component, through the use of a totally ordered broadcast communication service using a view-synchronous group communication service. The individual group communication requirements and an explanation thereof follows.

Property 1: “Self Inclusion”

Definition: “If process p installs view V , then p is a member of V .”

Formally: $installs(p, v) \Rightarrow p \in v.members$

The first required group communication property states that a process should always be a member of the view it is installing. In order to deliver a new group view, the vehicle should be part of the 4WS membership. It wouldn't make sense for a process to install a group view of which it was not a member. A group membership list is meant to be the set of processes with which a process can communicate. A process can always communicate with itself and as such should always be a member of any group it delivers.

Property 2: “Local Monotonicity”

Definition: “If a process p installs view V after installing view V' then the identifier of V is greater than that of V' .”

Formally:

$t_i = view_chng(p, v, T) \wedge t_j = view_chng(p, v', T') \wedge i > j \Rightarrow v.id > v'.id$

This property means that views are installed locally at any process in increasing order of view identifiers. This property guarantees that a member does not install the same view more than once and that if two members both install the same two views, they install these views in the same order. Local monotonicity is required because, otherwise, a vehicle could possibly install the same group view twice. This in turn could lead to duplicate messages being received from the GCS. Duplicate messages would result in multiple

updates to the shared data structure where only one was intended. This could lead to inconsistent views of the current traffic scenario and potentially lead to an accident.

Property 3: “Initial View Event”

*Definition: “Every **send**, **recv** and **safe_prefix** event occurs within some view.”*

Formally:

$$t_i = \text{send}(p, m) \vee t_i = \text{recv}(p, m) \vee t_i = \text{safe_prefix}(p, m) \Rightarrow \text{viewof}(t_i) \neq \lambda$$

This property means that all communication events, with respect to each process, must occur within a specific group view. This mainly relates to restricting a process from a **send** event before the first **view_chng** event. The advantage of such a requirement is in that a vehicle should not be allowed to send or receive shared data structure updates until it has joined the group and installed its initial group view. When a vehicle joins the primary component group, it is issued a copy of the shared data structure. Only then should the vehicle be in a position to update the shared data structure, otherwise, updates would be “blind”, in that the vehicle wouldn’t have a copy of the data structure which it is updating. Such blind updates could be meaningless and lead to an inaccurate shared data structure that doesn’t represent the true traffic situation.

Property 4: “Primary Component Membership”

Definition: “There is a one to one function f from the set of views installed in the trace to the natural numbers, such that f satisfies the following property:

for every view V with $f(V) > 1$ there exist a view V' , such that $f(V) = f(V') + 1$, and a member p of V that installs V in V' (i.e., V is the successor of V' at process p).”

Formally: $\exists f : \{v \mid \exists p : \text{installs}(p, v)\} \rightarrow N$ such that:

$$(f(v) = f(v') \Rightarrow v = v') \wedge \forall v (f(v) > 1 \\ \Rightarrow \exists v' (f(v) = f(v') + 1 \wedge \exists p \in v.\text{members} : \text{installs_in}(p, v, v')))$$

This property requires that for every pair of consecutive group views, there is a process that survives from the first group view to the second. This means that the intersection of the membership list of two consecutive group views is not the null set. The property of primary component membership is fundamental to a primary component membership 4WS implementation. From one group view to the next, at a minimum, a single vehicle must survive. Vehicles that expected to survive, but did not, will be expected to rejoin the primary group.

Property 5: “Delivery Integrity”

*Definition: “For every **recv** event there is a preceding **send** event of the same message.”*

Formally: $t_i = receive(p, m) \Rightarrow \exists q \exists j (j < i \wedge t_j = send(q, m))$

This property requires that messages are never generated spontaneously by the group communication service, that is every receive communication event must have a corresponding, preceding send communication event. If this requirement was not implemented, meaningless messages could be delivered to the 4WS application which would be undesirable if these messages were misinterpreted.

Property 6: “No Duplication”

*Definition: “Two different **recv** events with the same content cannot occur at the same process.”*

Formally: $t_i = recv(p, m) \wedge t_j = recv(p, m) \Rightarrow i = j$

Every message is received *at most once* by each member. The GCS should not deliver duplicate messages to the application. This implies that the GCS cannot offer the same quality of service as the underling network which could deliver duplicate messages to the member processes. If duplicate messages were allowed, then multiple data structure

updates could take place for a single intended update, which could lead to inconsistent views of the data structure within the primary component.

Property 7: “Same View Delivery”

Definition: “If a process p receives message m in view V , and some process q (possibly $p = q$) sends in view V' , then $V = V'$.”

Formally: $receives_in(p, m, v) \wedge receives_in(q, m, v') \Rightarrow v = v'$

If a GCS has the property of same view delivery, it guarantees that a message will be delivered in the same view at all members that deliver the message. The view in which the message is delivered need not necessarily be the same view in which the message was initially sent. With regards the 4WS, it is important that messages containing shared data structure updates should be delivered in the same view at each process in the group view. Otherwise, update messages will be received in different group views at different processes. This in turn would mean that local replicas of shared data structures would be inconsistent for periods of time. Inconsistent data structures implies that vehicles will have different views of the traffic situation, possibly two vehicles could both seize right of way and block each other on the junction.

Coupled with virtual synchrony and strong total ordering this property has the effect that all receiving group members can act upon shared data structure updates in a consistent manner. The result is that processes can take consistent actions based on received messages as their contexts will be identical.

Property 8: “Virtual Synchrony”

Definition: “If process p and q install the same new view V in the same previous view V' , then any message received by p in V' is also received by q in V' .”

Formally:

$$\begin{aligned} & \text{installs_in}(p, v, v') \wedge \text{installs_in}(q, v, v') \wedge \text{receives_in}(p, m, v') \\ & \Rightarrow \text{receives_in}(q, m, v') \end{aligned}$$

This property requires that two processes that participate in the same two consecutive views deliver the same set of messages in the former view. This property is useful for GCSs which operate in the presence of network partitions. Processes that remain connected will receive the same set of messages in the previous group view and hence will have identical contexts. Thus, these processes can continue updating, through message passing, the shared data structure. Disconnected processes (from the primary component) will be required to rejoin the primary group and therefore update their internal contexts. Only then can these disconnected processes continue updating the shared data structure

Property 9: “Safe Indication Prefix”

Definition: “If a message is indicated as safe, then it is stable in the view in which it was received”

Formally: $\text{indicated_safe}(p, m, v) \Rightarrow \text{stable}(m, v)$

All or nothing semantics are desirable in many distributed applications, however, in network environments where message loss is possible, all or nothing semantics are impossible to achieve. As an approximation to all or nothing, the concept of safe messages were introduced by the EVS Model [52]:

“A safe message m is received by the application at process p only when p 's GCS knows that the message is stable”, where, “a message is stable when all members of the current view have delivered this message to the application (and not just received the message from the network)”.

This property means that a message is indicated as safe only if it has been delivered to all member processes of the current view. Such an indication would be required in ensuring

that a shared data structure update has been carried out (delivered to the 4WS application) at all vehicles. The use of this property with totally ordered multicast messages would ensure consistent replication of the shared data structure among all member processes that deliver the messages.

Property 10: “Strong Total Order”

Definition: “There is a TS function f such that messages are received at all the processes in an order consistent with f .”

Formally:

$$\exists f(TS_function(f) \wedge \forall p \forall m \forall m' (recv_before(p, m, m') \Rightarrow f(m) < f(m')))$$

Strong total order multicast semantics require that messages are delivered in the same order at all processes that deliver these messages. These semantics will be used to maintain the consistency of replicated data in primary component groups, hence, will be used to maintain the consistency of the 4WS shared data structure in a primary component implementation. Strong total ordering only applies to processes that continue together in consecutive group views. Strong total ordering offers no guarantees to processes that are disconnected from the primary group. As such, again, disconnected processes will be expected to rejoin the primary group in order to receive totally ordered data structure updates.

The ten group communication requirements for a primary component-based 4WS application are summarised in table 5.3.

Property 1	Self Inclusion
Property 2	Local Monotonicity
Property 3	Initial View Event
Property 4	Primary Component Membership *
Property 5	Delivery Integrity
Property 6	No Duplication
Property 7	Same View Delivery
Property 8	Virtual Synchrony
Property 9	Safe Indication Prefix
Property 10	Weak Total Order *
* These properties are unique to a primary component implementation.	

Table 5.3: Summary of primary component requirements.

The advantage of a primary component 4WS implementation is that a connected majority of vehicles can always make progress. However, vehicles, not members of the primary component due to a network partition, will not be able progress and will be expected to rejoin the primary group. From the perspective of vehicles in the primary component, disconnected vehicles will be removed from the membership list and their corresponding shared data structure entries will also be removed using an “unexpected leave” primitive. In effect, disconnected vehicles will be viewed upon as “local vehicles”. In figure 5.3, part *a* shows a traffic situation before a network partition in which all vehicles are members of the same, primary, component. Part *b*, then shows the situation after a partition where, previously connected vehicles which are now disconnected are viewed upon as local vehicles. As can be seen, the primary component will be able to continue granting right of way to entitled vehicles as normal. This is because of an implicit knowledge that no other views of the traffic situation exist in, non-existent, non-primary components and that disconnected vehicles will be looked upon as local vehicles and handled accordingly. When disconnected vehicles rejoin the primary component they recover the state of the primary and form part of the new primary component.

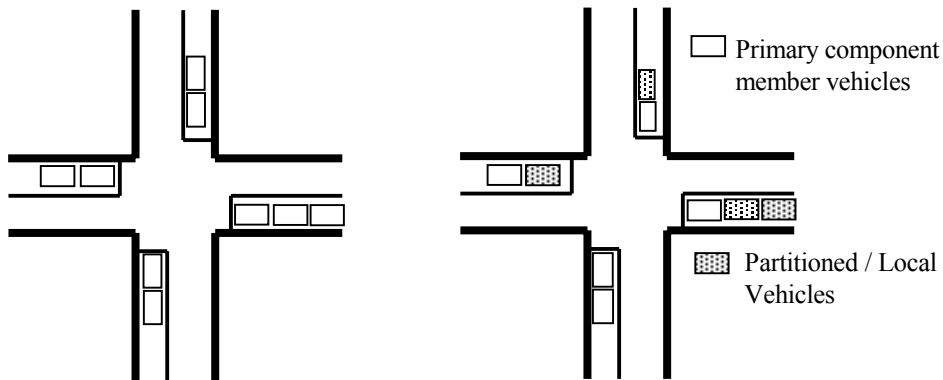


Figure 5.3a: A traffic scenario before partition.

Figure 5.3b: A traffic scenario after partition.

A notion of fairness could be used to argue that a primary component implementation would not respect the equal priorities of member vehicles and partitioned vehicles, treated as local vehicles. Treating a vehicle as a local vehicle implies that the vehicle is uninterested in gaining right of way at the junction, which is not necessarily the case with vehicles disconnected from the primary component. However, with safety being a paramount concern of any inter-vehicle coordination application, fairness must be a secondary concern.

Partitionable Membership Required Properties

A partitionable membership service allows different group views of the same group to co-exist. Such group views are called concurrent, disjoint group views. Network partitioning is a fact of life in most distributed systems, however, the presence of wireless links, as is the case in ad hoc networks, further increase the frequency and number of partitions. During a partition, multiple network components exist within a system. A partitionable membership service supports disconnected operation of these components i.e. components disconnected from each other are allowed to make progress by maintaining their own dynamic membership and by sending and receiving multicast messages. This is in contrast to a primary component membership service, which, requires only a single group view and hence only a single component, to exist within the system at any one time

With regards a 4WS partitionable membership implementation, groups of vehicles may be disconnected from each other during a network partition. These distinct vehicle groups

may continue to operate in the presence of such a partition by continuing to update their group membership and maintain their respective copies of the now disjoint, shared data structures. Group membership will be required to provide an ordered multicast service to each of these partitioned groups. Such a membership service, will be used to maintain the consistency of the shared data structures within each group. In a primary component implementation, a strong totally ordered multicast service was required. In a partitionable implementation a strong totally ordered multicast service is not relevant because it requires a single group view to exist in the system at any one time. However, a weak totally ordered multicast does not require such a constraint. It is upon this weak total ordering that the partitionable 4WS application will be built. A list of the required group communication properties for a partitionable membership service is now given.

Please, note that many of the group communication properties are required for both a primary component implementation and a partitionable membership implementation. As such, some of the following discussions are similar to discussions on the previous section.

Property 1: “Self Inclusion”

Definition: “If process p installs view V , then p is a member of V .”

Formally: $installs(p, v) \Rightarrow p \in v.members$

Again, this required group communication property states that a process should always be a member of the view it is installing, regardless of whether the system is partitioned or not. In order to deliver a new group view, the vehicle should be part of the 4WS membership. Because, a group membership list is meant to be the set of processes with which a process can communicate, and a process can always communicate with itself, it wouldn't make sense for a process to install a group view of which it was not a member.

Property 2: “Local Monotonicity”

Definition: “If a process p installs view V after installing view V' then the identifier of V is greater than that of V' .”

Formally:

$$t_i = \text{view_chng}(p, v, T) \wedge t_j = \text{view_chng}(p, v', T') \wedge i > j \Rightarrow v.id > v'.id$$

This property means that views are installed locally at any process in increasing order of view identifiers. This property guarantees that a member does not install the same view more than once and that if two members both install the same two views, they install these views in the same order. Local monotonicity is required because, otherwise, a vehicle could possibly install the same group view twice. This in turn could lead to duplicate messages being received from the GCS. Duplicate messages would result in multiple updates to the shared data structure where only one was intended. This could lead to inconsistent views of the current traffic scenario and potentially lead to an accident.

Property 3: “Initial View Event”

*Definition: “Every **send**, **recv** and **safe_prefix** event occurs within some view.”*

Formally:

$$t_i = \text{send}(p, m) \vee t_i = \text{recv}(p, m) \vee t_i = \text{safe_prefix}(p, m) \Rightarrow \text{viewof}(t_i) \neq \lambda$$

This property means that all communication events, with respect to each process, must occur within a specific group view. This mainly relates to restricting a process from a **send** event before the first **view_chng** event. The advantage of such a requirement is in that a vehicle should not be allowed to send or receive shared data structure updates until it has joined a group, be it a partitioned group or not, and installed its initial group view.

When a vehicle joins a group, it is issued a copy of the group’s shared data structure and added to the group’s membership list. Only then should the vehicle be in a position to update the group’s shared data structure, otherwise, updates would be “blind”, in that the vehicle wouldn’t have a copy of the data structure which it is updating. Such blind

updates could be meaningless and lead to an inaccurate shared data structure that doesn't represent the true traffic situation.

Property 4: "Delivery Integrity"

*Definition: "For every **recv** event there is a preceding **send** event of the same message."*

Formally: $t_i = receive(p, m) \Rightarrow \exists q \exists j (j < i \wedge t_j = send(q, m))$

This property requires that messages are never generated spontaneously by the group communication service, that is every receive communication event must have a corresponding, preceding send communication event. If this requirement was not implemented, meaningless messages could be delivered to the 4WS application which would be undesirable if these messages were misinterpreted.

Property 5: "No Duplication"

*Definition: "Two different **recv** events with the same content cannot occur at the same process."*

Formally: $t_i = recv(p, m) \wedge t_j = recv(p, m) \Rightarrow i = j$

Every message is received *at most once* by each member of a group, whether the group is within a partitioned component or a non-partitioned group. The GCS should not deliver duplicate messages to the application. This implies that the GCS cannot offer the same quality of service as the underlying network which could deliver duplicate messages to the member processes. If duplicate messages were allowed, then multiple data structure updates could take place for a single intended update, which could lead to inconsistent views of the data structure within a group (partitioned component or non-partitioned group).

Property 6: “Same View Delivery”

Definition: “If a process p receives message m in view V , and some process q (possibly $p = q$) sends in view V' , then $V = V'$.”

Formally: $receives_in(p, m, v) \wedge receives_in(q, m, v') \Rightarrow v = v'$

If a GCS has the property of same view delivery, it guarantees that a message will be delivered in the same view at all group members that deliver the message. The view in which the message is delivered need not necessarily be the same view in which the message was initially sent. With regards the 4WS, it is important that messages containing shared data structure updates should be delivered in the same view at each process in the group’s view. Otherwise, update messages will be received in different group views at different processes. This in turn would mean that local replicas of shared data structures, within groups, would be inconsistent for periods of time. Inconsistent data structures implies that vehicles will have different views of the traffic situation, possibly two vehicles could both seize right of way and block each other on the junction.

Coupled with virtual synchrony and weak total ordering this property has the effect that all receiving group members, within a partitioned component or non-partitioned group, can act upon shared data structure updates in a consistent manner. The result is that processes within specific groups can take consistent actions based on received messages as their contexts will be identical relative to the group of which they are members.

Property 7: “Virtual Synchrony”

Definition: “If process p and q install the same new view V in the same previous view V' , then any message received by p in V' is also received by q in V' .”

Formally:

*$installs_in(p, v, v') \wedge installs_in(q, v, v') \wedge receives_in(p, m, v')$
 $\Rightarrow receives_in(q, m, v')$*

This property requires that two processes that participate in the same two consecutive views deliver the same set of messages in the former view. This property is useful for GCSs which operate in the presence of network partitions. Whenever the network partitions, disconnected processes may diverge and reach different states. When disconnected processes merge, they must perform state transfer to reach a common state. Virtual synchrony allows transfer to be avoided among processes that continued together. The next property, transitional sets, is used to determine the subset of merged group members which will require state transfers.

With regards a 4WS implementation, during a network partition multiple disjoint views of the traffic situation may exist within the network. It must be possible for groups of vehicles to make progress during a network partition by continuing to update their respective shared data structures, although progress across the junction is not allowed during a partition. Once a network has merged, state transfer must take place in order for the previously disconnected groups of vehicles to once again establish a common view of the traffic situation and resume determining which vehicle has right of way.

Property 8: “Transitional Set”

Definition: “If process p installs a view V in (previous) view V' , then the transitional set for view V at process p is a subset of the intersection between the member sets of V and V' .”

,or,

“If two processes p and q install the same view, then q is included in p 's transitional set for this view if and only if p 's previous view was also identical to q 's previous view.”

Formally:

$$t_i = \text{view_chng}(p, v, T) \wedge \text{viewof}(t_i) = v' \Rightarrow T \subseteq v.\text{members} \cap v'.\text{members}$$

,or,

$$t_i = \text{view_chng}(p, v, T) \wedge \text{viewof}(t_i) = v' \wedge \text{installs_in}(q, v, v'') \Rightarrow (q \in T \Leftrightarrow v' = v'')$$

A transitional set contains information that allows processes to *locally* determine whether the hypothesis of virtual synchrony applies or a state transfer is required. When used in conjunction with virtual synchrony the transitional set delivered at process p reflects the set of processes whose states are identical to p 's state. In order to facilitate state transfers, it must be possible for processes to determine whether or not state transfers are actually required after a network partition has remerged. A possible solution would be for processes to piggyback their previous group membership details with any group communication messages sent. Processes could therefore locally determine if state transfer is required.

Groups of previously disconnected vehicles will use this property after a network merge to determine which vehicles will require state transfers. Updates to disjoint shared data structures (such as new vehicles joining) which took place during a network partition must be represented in the shared data structure of the remerged group. Transitional sets will be used to determine which vehicles will need to perform this state transfer, of shared data structure and membership list, to reach mutually consistent internal contexts. After a state transfer, weak totally ordered multicast and virtual synchrony will mean that vehicles that continue together in the same group view will continue to maintain their identical internal contexts and hence be able to act upon received messages in a consistent manner.

Property 9: "Safe Indication Prefix"

Definition: "If a message is indicated as safe, then it is stable in the view in which it was received"

Formally: $indicated_safe(p, m, v) \Rightarrow stable(m, v)$

Recall the definition of a safe message, introduced by the EVS Model [52], as an approximation to all or nothing semantics:

"A safe message m is received by the application at process p only when p 's GCS knows that the message is stable", where, "a message is stable when all members of the current

view have delivered this message to the application (and not just received the message from the network)”.

Again, the property of safe indication prefix means that a message is indicated as safe only if it has been delivered to all member processes of the current view. Such an indication would be required in ensuring that a shared data structure update has been carried out (delivered to the 4WS application) at all vehicles. The use of this property with weak totally ordered multicast messages would ensure consistent replication of the shared data structure among member processes that continue together through successive group and that actually deliver the messages.

Property 10: “Weak Total Order”

Definition: “For every pair of views V and V' there is a timestamp function f so that every process that installs V in V' receives messages in V' in an order consistent with f ”

, or,

“For every view V there is a timestamp function f so that every process that has V as its last view receives messages in V in an order consistent with f ”

Formally:

$$\forall V \forall V' \exists f ($$

$$TS_function(f) \wedge \forall p \forall m \forall m' (installs_in(p, v, v') \wedge recv_before_in(p, m, m', v')$$

$$\Rightarrow f(m) < f(m'))$$

, or,

$$\forall V \exists f ($$

$$TS_function(f) \wedge \forall p \forall m \forall m' (last_view(p, v) \wedge recv_before_in(p, m, m', v)$$

$$\Rightarrow f(m) < f(m'))$$

Weak total order semantics guarantee that processes that remain connected receive messages in the same order. Weak total order allows disconnected processes (i.e. processes in different network components due to a network partition) to disagree on the order of messages. However, if all processes remain permanently connected, then weak

total order offers the same guarantee as strong total order where all processes receive messages in the same order.

Weak total ordering in a 4WS implementation means that vehicles within a specific group (partitioned-component or non-partitioned group) receive the same shared data structure updates in the exact same order. This means that shared data structures within a group are consistently maintained, although, other versions of the shared data structure may exist in other network components.

The above properties, summarised in table 5.4, if guaranteed by a group communication service would provide a partitionable membership service 4WS application with the necessary tools to successfully support accurate, consistent and complete maintenance of the shared data structure.

Property 1	Self Inclusion
Property 2	Local Monotonicity
Property 3	Initial View Event
Property 4	Delivery Integrity
Property 5	No Duplication
Property 6	Sending View Delivery
Property 7	Virtual Synchrony
Property 8	Transitional Set *
Property 9	Safe Indication Prefix
Property 10	Weak Total Order *
* These properties are unique to a partitionable membership implementation.	

Table 5.4: Summary of partitionable membership service requirements.

The advantage of a partitionable membership 4WS implementation is in the fact that all network components can continue to make progress during a network partition. Each component maintains its own version of the 4WS shared data structure during a partition. Also, each component can continue to maintain its own dynamic vehicle membership.

Partitions are not hidden from the 4WS application, instead, the application becomes aware of network partitions and handles them by allowing group membership to change and allowing updates to the shared data structure. However, an exception to normal

operation is enforced during a partition: no vehicle is allowed to seize right of way during a network partition. If a vehicle in one partitioned component was to seize right of way and attempt to cross the junction, it cannot do so safe it the knowledge, that no other vehicle in a different partitioned component is also attempting to cross the junction. Thus, the seize right of way primitive is not allowed during a partition. Instead, network components will be required to fully merge before any vehicle can seize right of way. The act of merging will make use of the property of transitional sets to identify vehicles that must update their internal contexts (membership lists and shared data structure) to respect changes that occurred in other network components during a partition.

One possible disadvantage of not granting right of way during a network partition is that a partition may never remerge. However, according to the fault model, described above, we operate under the principle that every partition will eventually remerge.

This chapter offered group communication requirements to successfully maintain the consistency of the 4WS shared data structure. Requirements were listed for both a primary component implementation and a partitionable membership implementation. These requirements are not just specific to a 4WS implementation, they are applicable to any inter-vehicle coordination application, based on group communication, in which vehicles actions are determined by the contents of a shared data structure. The investigation of a specific inter-vehicle coordination scenario was used to identify the group communication requirements for inter-vehicle coordination applications in general. The next, and final, chapter will conclude the discussion.

Chapter 6: Conclusion

Advancements in wireless communication technology and portable computing have fuelled research in mobile ad hoc network applications. Ad hoc networks provide a unique application environment, in which applications are posed challenges in the form of node mobility, limited bandwidth availability, frequent network topology changes, network partitions and the absence of fixed infrastructure.

One application domain which must face these communication challenges is that of inter-vehicle coordination. Due to the mobility of vehicles, wireless communication, and hence ad hoc networks, provide the obvious communication domain to develop inter-vehicle coordination applications. This dissertation investigated the use of group communication to support the development of inter-vehicle coordination. We identified the requirements on group communication in a specific inter-vehicle coordination application: the 4 Way Stop.

By investigating this specific inter-vehicle coordination example, we made some important observations of inter-vehicle coordination applications in general.

Firstly, we believe that the notion of a proximity group will be important to any inter-vehicle coordination application. Vehicles' interests in coordinating their manoeuvres should not solely be based on a location aspect: it should also involve a functional aspect. This means that vehicles in the vicinity of an inter-vehicle coordination application need not necessarily be interested in coordinating their actions with other vehicles. For example, a coordination application to platoon vehicles, for efficient movement of traffic, should not try to involve parked vehicles, or emergency vehicles, which by their nature would be uninterested in "normal" traffic flow.

Core to the 4WS application was the use and maintenance of a shared data structure. From the perspective of individual vehicles, this shared data structure provided the basis on which vehicles decided upon their actions. From a wider perspective, the shared data structure was used to coordinate the actions of all interested vehicles within proximity of the junction. We believe that many, though perhaps not all, inter-vehicle coordination applications will involve the use of some form of shared data structure. Vehicles must decide upon their coordinated actions based on access to information regarding other involved vehicles and the traffic environment. We suggest that such information will be stored in the form of a shared data structure, which vehicles update and read from. Although, the complexity and size of the applicable shared data structures may vary: from a simple data structure to handle coordinated overtaking, to a intricate data structure to control coordination and navigation of vehicles through an urban setting, the notion of a shared data structure will be of paramount importance to inter-vehicle coordination.

An investigation to identify how group communication should support inter-vehicle coordination in an ad hoc network primarily involved determining the requirements on group communication to maintain the consistency of the shared data structure. An important observation was that the requirements on the consistency of the shared data structure varied depending on the implementation of choice: primary component implementation or partitionable membership implementation. In a partitionable implementation multiple versions of a shared data structure can exist concurrently within a system (one version per partitioned components), whereas in a primary component only one version can exist at any time. Again, depending on the implementation of choice, an inter-vehicle application may have to constrain application primitives, as both primary component and partitionable membership implementations did in the face of network partitions. Recall, for example, that in a partitionable membership implementation, vehicles could not seize right of way to cross the junction during a network partition. Other such application constraints may be necessary in other inter-vehicle coordination application.

In chapter five, we presented the requirements on group communication to maintain a 4WS shared data structure in a partitionable ad hoc network. These requirements were presented in two classes: those for a primary component implementation and those for a

partitionable membership implementation. These requirements made no stipulations on the type of, or contents of, the shared data structure. This means that these requirements can be used by any inter-vehicle coordination application to maintain a shared data structure, indeed, the requirements are applicable to any data replication application in a partitionable network. Also presented was a fault model for these requirements. This fault model must be addressed by any inter-vehicle coordination application operating in an ad hoc network.

As was the case in the 4WS, we foresee that in inter-vehicle coordination, group membership and the associated shared data structure will be strongly linked, in that every group member may have an associated entry in the shared data structure. As such, an inter-vehicle coordination application may need to respect certain real world physical conditions. For example, in the 4WS application a physical ordering requirement was stipulated to ensure the shared data structure represented the actual physical ordering of vehicles on an approach. Standard group communication “join” primitives offer no such respect for a mobile nodes physical location: a node which joins a group is simply added to the next group view. We foresee that location information, through GPS for example, will have to be used in conjunction with group communication primitives to respect such physical constraints.

Obviously, the primary use of this research is as a reference for an implementation of a 4WS application. Although specific in its examination of the 4 Way Stop case study, the above conclusions and observations and indeed, the entire dissertation, will prove useful in the development of general inter-vehicle coordination applications. The specified requirements on group communication to successfully maintain an inter-vehicle coordination shared data structure are also applicable to all such applications.

Future Work

The most obvious suggestion for future work would be the implementation, or simulation, of a 4WS group communication-based protocol based on the group communication requirements and shared data structure proposed by this research. Perhaps, two versions of a 4WS implementation could be developed: a primary component implementation and a partitionable membership implementation. Then these two implementations could be

tested and analysed to determine which implementation works most efficiently in terms of both group communication cost and application level costs. Such research would further provide an insight into group communications support of inter-vehicle coordination.

Although, not directly related to group communication, other research could be carried out to determine a suitable 4WS shared data structure and right of way algorithm to maximise performance or fairness at the 4WS junction. This suggested research is fuelled by the implicit insight that at a junction, there is a fixed overhead, in terms of time, for every vehicle to cross the junction: the vehicle must accelerate from the front of the approach, turn (or go straight), and exit the junction before another vehicle can attempt to cross the junction. Perhaps, groups of vehicles, from the same approach, should cross the junction “simultaneously” to maximise throughput at the junction. Otherwise, perhaps, the right of way algorithm could be optimised for fairness, such that vehicles longest queuing on an approach, and not just the front of the approach as is the case with the 4WS, are prioritised over vehicles queuing for shorter periods of time.

References

- [1] K. Birman, “The process group approach to reliable distributed computing.”, Communications of the ACM, December 1993.

- [2] K. Birman, “Building Secure and Reliable Network Applications.”, chapters 13-18, Manning Publications, 1996.

- [3] K. Birman and R. van Renesse, “Reliable Distributed Computing with the Isis Toolkit.”, IEEE Computer Society Press, 1994.

- [4] K. Birman, “ISIS: A System for Fault-Tolerant Distributed Computing.”, Technical Report , Department of Computer Science, Cornell University, April 1986.

- [5] R. Friedman and A. Vaysburg, “Fast replicated state machines over partitionable networks.”, In 16th IEEE International Symposium on Reliable Distributed Systems, October 1997.

- [6] R. Friedman and A. Vaysburg, “High-performance replicated distributed objects in partitionable environments.”, Technical Report 97-1639, Department of Computer Science, Cornell University, Jul 1997.

- [7] I. Keidar and D. Dolev, “Efficient message ordering in dynamic networks.”, In 15th ACM Symposium on Principles of Distributed Computing, May 1996.

- [8] R. Khazan, "Group Communication as a base for a load-balancing, replicated data service.", Masters thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 1998.
- [9] R. Khazan, A. Fekete and N. Lynch, "Multicast Group Communication as a Base for a Load-balancing Replicated Data Service.", In 12th International Symposium on Distributed Computing, September 1998.
- [10] A. Schiper and M. Raynal, "From group communication to transactions in distributed systems.", In Communications of the ACM, April 1996.
- [11] B. Kemme and G. Alonso, "A suite of database replication protocols based on group communication primitives.", In 18th International Conference on Distributed Computing Systems, June 1999.
- [12] J. Sussman and K. Marzullo, "The Bancomat problem: An example of resource allocation in a partitionable asynchronous system.", In 12th International Symposium on Distributed Computing, September 1998.
- [13] O. Babaoglu, R. Davoli, A. Montresor and R. Segala, "System Support for Partition-Aware Network Applications.", In 18th International Conference on Distributed Computing Systems, May 1998.
- [14] S. Mishra, and G. Pang, "Design and implementation of an availability management service.", In 19th International Conference on Distributed Computing Systems Workshop on Middleware, June 1999.
- [15] G. Shamir, "Shared Whiteboard: A Java Application in the Transis Environment.", Lab project, High Availability Lab, The Hebrew University of Jerusalem, 1996.
- [16] M. Valenci, "Audio Conferencing using Transis.", Lab project, High Availability Lab, The Hebrew University of Jerusalem, 1998.

- [17] S. Chodrow, M. Hirsch, I. Rhee and S. Cheung, "Design and implementation of a multicast audio conferencing tool for a collaborative computing framework.", In JCIS, March 1997.
- [18] E. Al-Shaer, A. Youssef, H. Abdel-Wahab, K. Maly and C. Overstreet, "Reliability, scalability and robustness issues in IRI.", In IEEE Sixth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, June 1997.
- [19] Linda Briesemeister, "Group Membership and Communication in Highly Mobile AdHoc Networks.", Technical University of Berlin, School of Electrical Engineering and Computer Sciences, Berlin, November, 2001.
- [20] R. van Renesse, K.P. Birman and S. Maffeis, "Horus: A Flexible Group Communication System.", In Communications of the ACM, vol.39, no.4, April 1996.
- [21] D. Agarwal, R. Budhia and C. Lingley-Papadopoulos, "Totem: A Fault-Tolerant Multicast Group Communication System.", In Communications of the ACM, 39 (4) April 1996.
- [22] Y. Amir, D. Dolev, S. Kramer, D. Malki, "Transis: A Communication Sub-System for High Availability.", In 22nd International Conference on Fault-Tolerant Computing, pp.76-84, IEEE Computer Society Press, 1992.
- [23] G.V. Chockler, I. Keidar, R. Vitenberg, "Group Communication Specifications: A Comprehensive Study", In ACM Computing Surveys 33(4), pages 1-43, December 2001.
- [24] K. Birman and T. Joseph, "Exploiting Virtual Synchrony in Distributed Systems.", In 11th ACM SIGOPS Symposium on Operating Systems Principles, November 1987.
- [25] H. Tan, R. Rajamani and W. Zhang, "Demonstration of an automated highway platoon system," American Control Conference (ACC), 1998.

[26] K. Chang, W. Li, A. Shaikhbahai and P. Varaiya, "A preliminary implementation for vehicle platoon control system", In Proceedings of the 1991 American Control Conference, June 1991.

[27] W. Chee and M. Tomizuka, "Lane change manoeuvre of automobiles for the intelligent vehicle and highway systems", In Proceedings of American Control Conference, 1994.

[28] "The Global Positioning System", The Aerospace Corporation, Los Angeles, 1999.

[29] E. Blackwell, "Overview of differential GPS methods.", In Papers published in Navigation, vol. 3, pp. 89-100, 1980.

[30] Christopher K. H. Wilson, Seth Rogers, and Shawn Weisenburger, "The Potential of Precision Maps in Intelligent Vehicles.", IEEE International Conference on Intelligent Vehicles, pages 419-422, October 1998.

[31] S. Rogers and S. Schroedl, "Creating and Evaluating Highly Accurate Maps with Probe Vehicles" IEEE Conference on Intelligent Transportation Systems, Dearborn, MI, USA, October 2000.

[32] S. Bohrer, M. Brauckmann and W. von Seelen, "Visual Obstacle Detection by a Geometrically Simplified Optical Flow Approach", In 10th European Conference on Artificial Intelligence Proceedings, 1992.

[33] N. Ancona, "A Fast Obstacle Detection Method based on Optical Flow.", In Proceedings of the European Conference on Computer Vision, 1992.

[34] J. Bruyelle and J. Postaire, "Direct Range Measurement by Linear Stereovision for Real-Time Obstacle Detection in Road Traffic.", In Proceedings of the International Conference on Intelligent Autonomous Systems, 1993.

[35] S. Cornell, J. Porrill and J. Mayhew, "Ground Plane Obstacle Detection Under Variable Camera Geometry Using a Predictive Stereo Matcher.", In Proceedings of the

British Machine Vision Conference, 1992.

[36] C. Frölich, M. Mettenleiter and F. Haertl, “Imaging Laser Radar for High-Speed Monitoring of the Environment.”, In Proceedings of the SPIE Conference on Intelligent Transportation Systems, 1997.

[37] J. Hancock, M. Hebert, and C. Thorpe. “Laser Intensity-Based Obstacle Detection.”, In Proceedings of the IEEE Conference on Intelligent Robots and Systems, 1998.

[38] J. Hancock, E. Hoffman, R. Sullivan, D. Ingimarson, D. Langer and M. Hebert, “High performance laser range scanner.”, In Proceedings of the SPIE Conference on Intelligent Transportation Systems, 1997.

[39] D. Langer, “An Integrated MMW Radar System for Outdoor Navigation.”, Ph.D. Thesis, Carnegie Mellon Technical Report, CMU-RI-97-03, 1997.

[40] Y. Amir, D. Dolev, S. Kramer, D. Malki, “The Transis Approach to High Availability Cluster Communication.”, In Communications of the ACM, 39 (4) April 1996.

[41] G.-C. Roman, Q. Huang and A. Hazemi, “Group Membership Data in Ad Hoc Networks”, Washington University, St. Louis, technical report wucs-00-26, April 16, 2000.

[42] M.-O. Killijian, R. Cunningham, R. Meier, L. Mazare and V. Cahill, “Towards Group Communication for Mobile Participants.”, Proceedings of the 1st ACM Workshop on Principles of Mobile Computing (POMC), Newport, Rhode Island, USA, August 29-30, 2001.

[43] R. Prakash and R. Baldoni, “Architecture for Group Communication in Mobile Systems”, Symposium on Reliable Distributed Systems, West-Lafayette (IN), USA, 1998.

[44] W.J. Franz, H. Hartenstein, B. Bochow, “Internet on the Road via Inter-Vehicle Communication”, DaimlerChrysler AG, NEC Europe Ltd., GMD FOKUS.

[45] <http://www.rtna.daimlerchrysler.com>

[46] “DriveBy InfoFueling - Telematics beyond the Anytime Anywhere Paradigm” whitepaper, Mercedes-Benz USA, Montvale, New Jersey; DaimlerChrysler Research and Technology North America, Palo Alto, California, November 2001.

[47] Robert Morris, John Jannotti, Frans Kaashoek, Jinyang Li and Douglas S. J. De Couto, “CarNet: A Scalable Ad Hoc Wireless Network System.”, Proceedings of the 9th ACM SIGOPS European workshop: Beyond the PC: New Challenges for the Operating System, September 2000, Kolding, Denmark.

[48] Min-te Sun, Wu-chi Feng, Ten-Hwang Lai, Kentaro Yamada, Hiromi Okada, Kikuo Fujimura “GPS-Based Message Broadcasting for Inter-vehicle Communication.”, Proceedings of the Proceedings of the 2000 International Conference on Parallel Processing, 2000.

[49] P. Varaiya, “Smart cars on smart roads: Problems of control,” IEEE Transactions on Automatic Control, vol. 38, Feb 1993.

[50] “Intelligent vehicle highway systems”, Special Issue on Vehicle System Dynamics, vol. 26, no. 4, 1996.

[51] P. Varaiya, “Models, simulation, and performance of fully automated highways”, California PATH Research Report, 1994.

[52] L. Moser, Y. Amir, P. Melliar-Smith and D. Agarwal, “Extended Virtual Synchrony”, In 14th International Conference on Distributed Computing Systems, 1994.

[53] L. Lamport, R. Shostak and M. Pease, “The Byzantine Generals Problem.”, In ACM Transactions on Programming Languages and Systems 4, 1982.