# Supporting Group Communication in Mobile Ad-Hoc Networks

**Kulpreet Singh**

A thesis submitted to the University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

Doctor of Philosophy

October 2006

# Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work.

Kulpreet Singh

Dated: October 27, 2006

## Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Kulpreet Singh

Dated: October 27, 2006

To Ammi and Bigoo

# Acknowledgements

I would like to express my deep gratitude to my supervisor, Dr. Siobhán Clarke for the invaluable guidance and support during the last five years of training. The invaluable lessons that I learnt from her on cross examining my own thoughts and ideas will surely be some of the most important lessons of my life. I have been lucky to have her patience and understanding while I floundered in the dark.

I would like to thank Prof. Vinny Cahill for the his valuable time that he spent with me, and most importantly to give me the opportunity to work in the area of distributed systems. I would also like to extend a special thanks to Dr. Stefan Weber for providing guidance during periods of uncertainty.

Thanks to Siobhán and Vinny will be incomplete without saying how grateful I am for all the help they provided me during my troubles. Thanks a gazillion.

While working with DSG I have been extremely lucky to meet and befriend some of the most enriching individuals. I'd like to thank Andronikos Nedos for providing endless k-debates[1] and support during times only the two of us will know about (Maybe Zelator knows too). A big thanks to Stefan Weber, Raymond Cunningham and Mads Haahr for support, encouragement and for being people with golden hearts. Although not in DSG, Christine Appel deserves a special thanks, for being a friend and for "marshalling forces" to help me out when I need them most.

DSG has given so many friends that it is impossible to list them all, so thanks to anyone who ever had a beer (or played a game of starcraft) with me and heard me moan about the fish and everything. Those sessions were much needed. Thanks.

[1] Debating ideas that could not be conclusive proved or disproved.

Finally, I'd like to thank my parents for being patient whenever I said "one more year". My sisters, Akrit and Prabhmeet, for somehow being there for me all through these five years, even if I always was the lazy one trying to stay in touch across the seas. And a special thanks to my Dadima for encouraging me to study further right when I most needed the encouragement.

A special thanks is also due to Carla DeTona for having the patience and understanding while I did my elephant walk in a crystal shop. You provided the pillar in Dublin where I could work from.

**Kulpreet Singh**

*University of Dublin, Trinity College*

*October 2006*

# Abstract

Mobile Ad-Hoc Networks (MANETs) are networks formed when mobile nodes communicate over a wireless medium as and when they come within each other's radio range. Highly unstable communication links are characteristic of MANETs, and result in networks frequently partitioning and merging. Such an environment makes it difficult to provide reliable communication guarantees to applications.

Research in the area of reliable communications for fixed communication networks like LANs and WANs has resulted in a number of protocols and systems that provide varying degrees of reliable communication guarantees. However, solutions from LANs and WANs can not be used in MANETs, not only because of frequent partitions and mergers in MANETs, but also because the communication protocols used in MANETs are very different from those used in fixed networks.

Nonetheless, there is a significant benefit to be gained from considering the paradigms that work for fixed networks, in particular Group Communication Services (GCS). A GCS is useful for developing distributed applications with reliable communication guarantees as it provides all participating nodes with a list of other nodes with whom they can reliably communicate, this list is called a group view. Further, a GCS provides a reliable many to many communication facility between the nodes in a group view.

A GCS service for MANETs that provides consistent group views and reliable communication guarantees is a problem that has not yet been fully solved. This work describes TransMAN, a GCS for MANETs that provides consistent group views to all participating nodes and also provides a reliable FIFO and total ordered many-to-many communication facility between nodes in a group view.

TransMAN models a MANET as a number of overlapping broadcast domains with randomly changing gateways that connect these broadcast domains. This thesis presents protocols that provide reliable ordered broadcast and membership agreement over these composite broadcast domains. The membership agreement protocol uses broadcasts over the composite broadcast domains to provide a membership service that allows group partitions and mergers.

To date, research into reliable communication protocols in MANETs has focussed primarily on optimisations to improve the efficiency and reliability of broadcasts. These optimisations reduce the number of broadcasts required to disseminate messages in a MANET and try to increase the reliability of these broadcasts. In contrast to the research effort for optimising broadcasts, much smaller effort has gone into group view management for MANETS. The approaches that have been considered for group view management in MANETs range from providing participating nodes with a partial set of nodes, to assuming synchronous communication models for MANETs. None of these approaches provide a deterministic group view management service or reliable many-to-many communication.

TransMAN builds on the work of broadcast optimisations to provide a reliable broadcast communication service and combines this with a group management service to deliver a complete GCS for MANETs. The overhead of the membership service is studied by comparing average message delivery latency and the traffic generated with and without the membership service. The reactivity of the membership service is also studied to evaluate how quickly a new group view is installed when nodes fail or new nodes join the network. The evaluation shows TransMAN introduces only a small overhead in terms of number of messages and their delivery latencies even in the face of node mobility. Node mobility also does not increase the reaction time of TransMAN's membership service.

## Publications Related to this Ph.D.

1. Kulpreet Singh, Andronikos Nedos, Gregor Gaertner and Siobhán Clarke, Message Stability and Reliable Broadcasts in Mobile Ad-Hoc Networks, ADHOC-NOW, October 2005, 297-310, V.R. Syrotiuk and E. Chávez, LNCS 3788.

2. Kulpreet Singh, Andronikos Nedos and Siobhán Clarke, TransMAN: A Group Communication System For MANETs, ICDCN, December 2006, 430-441, S. Chaudhuri, S. R. Das, H. S. Paul, S. Tirthapura, LNCS 4308.

3. Kulpreet Singh, On Group Communication, Congestion Avoidance and Node Starvation in MANETs, Feburary 2006, Third Minema Workshop, Belgium.

# Contents

# List of Figures

# List of Tables

# List of Pseudocode

# Chapter 1

# Introduction

This thesis presents TransMAN, a group communication system for mobile ad-hoc networks. TransMAN provides a reliable broadcast with ordered message delivery along with a group membership service. Message delivery is provided in both FIFO and total orders and the membership service handles the partition and merger of MANETs. This chapter presents the motivation for a reliable communication service in a MANET and lists the main contributions of this thesis. Finally, a brief overview of the evaluation strategy is presented followed by the road map for the thesis.

## 1.1 Motivation

Mobile Ad-Hoc Networks or MANETs are formed whenever mobile devices start communicating over a wireless communication channel in an impromptu manner. Such a network formation creates opportunities for supporting applications such as collaborative work, mobile multiplayer games or simply applications that use resources in the vicinity of mobile nodes. These opportunities in ad-hoc networks come at a cost that communication protocols need to handle network formations and partitions without prior warnings. Such requirements are currently not satisfied in MANETs, and this precludes the development of applications that require reliable communication between mobile nodes.

Some simple applications in MANETs, such as those that locate and use one other

device like a printer or monitor, require only one-to-one reliable communication between mobile nodes. Routing protocols like AODV (Perkins, Belding-Royer & Das 2003), DSR (Perkins 2001), ODMRP (Lee, Su & Gerla 2000) or ZRP (Haas & Pearlman 1999) are designed to address the problem of enabling one-to-one communication in a MANET.

However, other more complex applications need many-to-many reliable communication between devices. For example, applications like multiplayer games or collaborative work applications require reliable communication between all devices participating in the mobile game or the collaborative work. Such applications require guarantees that (1) participating node have knowledge of other participating nodes and (2) messages are reliably delivered to all participating nodes in a known order. These requirements have been studied as many-to-many reliable communication guarantees (Mullender 1989, Chapter 5) and this thesis focusses on providing the guarantees in MANETs.

A many-to-many reliable communication with the above guarantees entails delivering messages from all participating nodes to all others even in the presence of message loss or corruption and node failures. If a node does not receive a message, the communication system has to resend the message to the node. To determine if a node has received a message, reliable communication systems gather acknowledgements of messages from participating nodes. If a node does not acknowledge a message the system knows the node has not received a message.

To gather acknowledgements and to determine if any of the participating nodes have not delivered a message the reliable communication system has to know which nodes are present in a reliable communication. In a one-to-one communication this is achieved by determining if there is a route to the other node or not. In many-to-many reliable communication a list of participating nodes is maintained. This list is called a "membership view" (Babaoğlu, Davoli & Montresor 2001, Chockler, Keidar & Vitenberg 2001, Hiltunen & Schlichting 1995). The membership view may be static if no new nodes join or leave the communication. Otherwise, a membership service must maintain an accurate membership view reflecting node joins and leaves. Maintaining a consistent membership view at all participating nodes provides for the delivery of messages in the same view to all nodes.

The maintenance of consistent views in the face of frequent view changes caused by the dynamic nature of MANETs implies a membership service should not block applications from sending or receiving messages while a group view changes. If applications block whenever a view change is taking place very little communication will be possible in MANETs where frequent view changes are expected. The approach of non-blocking view changes is in contrast to the approaches taken in early systems for fixed communication networks where applications are blocked during a group view's change (Birman, Constable, Hayden, Kreitz, Rodeh, van Renesse & Vogels 2000). However allowing applications to send messages while a group view change is taking place results in the application being unsure of which messages were delivered by other nodes during the group view change. To eliminate this uncertainty the property of "virtual synchrony" was introduced by Birman & Joseph (1987a).

A group communication system providing virtual synchrony guarantees that nodes that go through the same sequence of changes in group view deliver the same set of messages. Supporting virtual synchrony in MANETs is important because of frequent membership changes caused by the dynamic nature of MANETs. The group membership service described in this thesis provides virtual synchrony while allowing applications to make progress while a group view change is taking place.

In conclusion the reliable communication guarantees that this thesis focusses on are:

R-I Reliable Broadcast – A reliable broadcast guarantees that if one node receives a message all participating nodes receive the message; no message are lost; and no spurious messages are delivered to any node (Mullender 1989, Chapter 5).

R-II Ordered Delivery of Messages – Messages are delivered in well known order, like FIFO or total order.

R-III Consistent list of participating nodes – All participating nodes have the same view on the set of nodes participating in the reliable communication.

R-IV Non-Blocking View Changes – Applications should not be blocked from sending or receiving messages while a group view change is taking place.

R-V Virtual Synchrony – Nodes that survive the same group view changes deliver the same set of messages.

The next section briefly presents the existing work on systems for LANs and MANETs that strive to provide these guarantees.

## 1.2 Group Communication

Systems that provide both a reliable communication facility and a membership service are called group communication systems, or GCS. Group communication systems are used to provide reliable communication guarantees in fixed networks (LANs and WANs) and ease the development of distributed application (Birman & Joseph 1987b). GCS have even been implemented by various operating systems to support distributed applications (Cheriton & Zwaenepoel 1985, Kaashoek, Tanenbaum & Verstoep 1993). Efforts in implementing group communication systems for fixed networks have resulted in a plethora of group communication systems (Keidar, Sussman, Marzullo & Dolev 2002, Das, Gupta & Motivala 2002, van Renesse, Maffeis & Birman 1996, Cristian & Schmuck 1995, Ezhilchelvan, Macedo & Shrivastava 1995, Babaoğlu & Schiper 1995a, Kaashoek et al. 1993, Amir, Dolev, Kramer & Malki 1992, Schiper, Birman & Stephenson 1991, Cristian 1991, Cheriton & Zwaenepoel 1985, Amir, Moser, Melliar-Smith, Agarwal & Ciarfella 1995, Dolev & Malki 1995, Moser, Melliar-Smith & Agarwal 1994, Birman & Joseph 1987b).

The work on implementing group membership services for fixed networks has brought a formal understanding of their behaviour, resulting in a taxonomy of properties that can be used to characterise group communication systems. These properties, presented independently by Babaoğlu et al. (2001), Chockler et al. (2001), and Fekete, Lynch & Shvartsman (2001), can be used to specify a group communication system. The understanding reached while developing group communication systems for fixed networks helps us in providing a group communication system for dynamic networks like MANETs.

| Challenge | Implication |
|---|---|
| Frequent Topology Changes | A node's neighbours are frequently changing, resulting in a change in the path taken by messages. This requires protocols that are independent of topology information because gathering topology information in MANETs leads to extraneous communication messages. |
| Lack of Collision Detection | In IEEE 802.11b a node has no way to determine if a message transmission has been successful. A message sent by a node can collide with another message transmission making both the message transmissions redundant. This requires protocols that reduce the number of possible collisions. |
| Limited Batteries | Each message transmission over wireless radio channel is an expensive operation and the limited batteries on mobile devices requires minimising the number of message transmissions. |
| Transient Network Connections | Nodes reach each others radio range for a small period of time, transiently connecting with each other and triggering membership view changes. If the membership service blocks the application from sending or receiving messages during these transient connectivities, applications in MANETs will be frequently blocked. Protocols are required which allow applications to continue sending and receiving messages while MANETs partition or merge. |

Table 1.1: Challenges in MANETs

### 1.2.1 Group Communication for MANETs

Group communication systems for MANETs have lately received some attention (Briesemeister & Hommel 2002, Viswanath & Obraczka 2002, Luo, Eugster & Hubaux 2004, Friedman 2003, Meier, Killijian, Cunningham & Cahill 2001, Roman, Huang & Hazemi 2001). The dynamic nature of MANETs and the unreliability of the communication medium pose new challenges for implementing a group communication system. Further, the differences in the communication protocols used in MANETs and fixed networks discourage the use of solutions from the fixed networks as discussed now.

Some key differences between fixed networks and MANETs' communication mediums are (1) the use of wireless radio with communication links that are highly susceptible to failure, (2) node mobility, (3) high rate of collisions in wireless communication and (4) small

battery capacities of mobile devices. These differences create new challenges in developing communication systems for MANETs and preclude the direct deployment of solutions from fixed networks. The challenges this thesis explicitly focusses on are (1) frequently changing topologies, (2) lack of collision detection in MANET communication protocols, (3) limited battery power for MANETs and (4) transient network connections. Table 1.1 lists the challenges and their implications in providing reliable communication for MANETs.

To handle the challenges listed in Table 1.1 current group communication systems for MANETs use probabilistic approaches or gossip based approaches (Luo et al. 2004, Jo, Eugster & Hubaux 2003). These approaches help reduce the number of message transmissions but the group view provided is only a random subset of the nodes present in the communication. Further, the random views provided at each participating node is inconsistent at all these nodes. Such inconsistent and random lists of participating nodes make it hard to develop distributed applications like mobile multiplayer games or collaborative work. Other approaches that are tried for MANETs include making assumptions of a synchronous network (Roman et al. 2001) and using a coordinator to provide a fuzzy membership service (Friedman 2003). The diversity of devices in a MANET make the synchronous network assumption impossible to provide while the solution for implementing the fuzzy group membership uses a coordinator which will need frequent re-elections in a dynamic network like a MANET. A detailed discussion on these issues is presented in Chapter 2 on Related Work. This thesis presents, TransMAN, a group communication service that addresses the challenges presented by MANETs. Table 1.2 shows how our system handles these challenges.

## 1.3 MANETs as Broadcast Domains

Handling the challenges in MANETs requires a solution that utilises the inherent strengths of the communication model while handling its weaknesses. The strength of a wireless communication protocol like IEEE 802.11b is the broadcast nature of the communication. When a node sends a message it is broadcast in the region around it, and a good solution for MANETs will take advantage of this broadcast medium. The major weakness of wireless

| Challenge | How tackled |
|---|---|
| Frequent Topology Changes | The protocols described in this thesis do not depend on or maintain any topology information. |
| Lack of Collision Detection | The reliable broadcast protocol described in this thesis uses implicit acknowledgements to reduce transmissions and uses random timeouts (apart from those in the MAC) to reduce collisions. |
| Limited Batteries | Reducing the number of message transmissions is the only way to handle limited battery life and TransMAN uses the broadcast nature of wireless communication medium to disseminate a message to as many receiving nodes as possible. Use of implicit acknowledgements also reduces these transmissions. |
| Transient Network Connections | The solution described allows view changes without blocking application progress. Further, the simultaneously tracking of a number of potential view changes allows applications to be aware of other nodes which are in the network but are not yet members of the group. |

Table 1.2: Handling the Challenges

communication protocols is the potential increase in collisions when a number of nodes broadcast messages in the same geographical region.

Reducing the number of potential collisions in a MANET is best achieved by reducing the number of message transmissions in the network and also by using informed backoff schemes. These techniques are under intensive study by the MANET research community who have generated a number of mature and robust schemes to address this weakness of wireless communication protocols (section 2.2.3 on page 18 looks at these techniques). TransMAN is designed to work with any of these techniques.

To take advantage of the broadcast based communication available in wireless communication protocols, especially IEEE 802.11b, TransMAN considers a MANET as composed of a number of broadcast domains. Each broadcast domain surrounds a node and extends until the end of its radio range. Figure 1.1(a) shows a node $p$'s broadcast domain and the extent of the broadcast domain is shown in grey.

In a MANET each node has zero or more nodes in its broadcast domains. These nodes, called neighbours, in turn have other nodes in their broadcast domains. In this thesis a

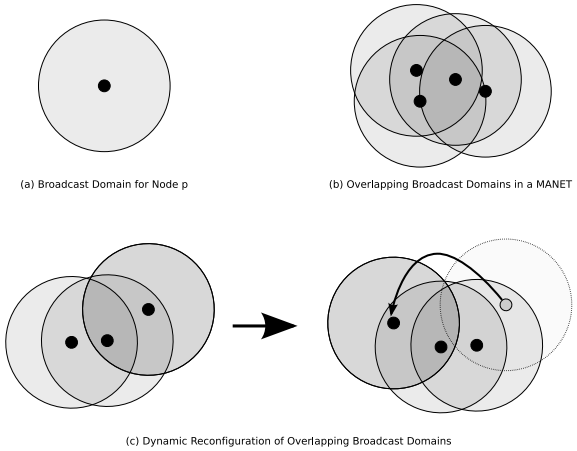


Figure 1.1: Broadcast Domains in MANETs

MANET is considered to be composed of a number of overlapping broadcast domains. Figure 1.1(b) shows a MANET consisting of a number of broadcast domains.

Group communication systems have been studied for broadcast domains where the broadcast medium is highly reliable. For example Psync (Peterson, Buchholz & Schlichting 1989), Trans (Melliar-Smith & Moser 1993) and Transis (Dolev & Malki 1995) are broadcast protocols and group communication systems studied for the broadcast domain of an Ethernet. Section 2.3.1 surveys the most prominent systems that take advantage of a broadcast domain to implement a group communication system. Some of these systems connect a number of broadcast domains via specially designated gateways to provide a group communication system over a WAN.

The approach of connecting a number of broadcast domains using designated gateways can not be used in MANETs because node mobility results in broadcast domains being connected by a different set of nodes during a period of execution. Figure 1.1(c) shows how mobility of a node results in broadcast domains being connected via different set of nodes. The broadcast domains connection through a dynamic gateways precludes the use of solutions from fixed networks that use designated gateways to maintain ordering and membership information.

TransMAN models a MANET as a set of overlapping broadcast domains without any designated gateways between them. The TransMAN approach is influenced by the work

by Peterson et al. (1989), Melliar-Smith & Moser (1993), Dolev & Malki (1995), Mishra, Peterson & Schlichting (1993) and Amir et al. (1995). These researchers targeted highly reliable broadcast domains like Ethernet. In contrast, TransMAN deals with the challenges of wireless communication and the absence of designated gateways between the wireless broadcast domains.

## 1.4 Thesis Contributions

As discussed earlier in this chapter a reliable communication system should provide the following guarantees: (R-I) Reliable broadcast, (R-II) Ordered delivery of messages, (R-III) Consistent list of participating nodes, (R-IV) Non-blocking view changes and (R-V) Virtual synchrony. Providing these guarantees in MANETs is useful for developing applications with reliable communication guarantees, but there are problems and challenges in MANETs that need to be addressed while developing such a system. These challenges of (1) frequent topology changes, (2) lack of collision detection, (3) limited batteries and (4) transient network connections were discussed in Table 1.1 and the approach taken by TransMAN in this thesis is described in Table 1.2. The contributions of this thesis are:

Ordered Reliable Broadcast for MANETs – The current work on broadcast protocols in MANETs (Pagani & Rossi 1997, Wan & Campbell 2002, Lou & Wu 2004, Vollset & Ezhilchelvan 2005) does not provide ordered message delivery which puts the onus of ordering messages on a higher layers in the system. TransMAN's reliable broadcast provides a FIFO and total ordered message delivery which can work with any of the existing broadcast optimisations.

Consistent Membership Views in MANETs – Current work on providing group views in MANETs either make assumptions which are hard to realise in MANETs (Roman et al. 2001), or provide probabilistic and inconsistent membership views (Bar-Yossef, Friedman & Kliot 2006, Friedman 2003, Luo et al. 2004, Meier et al. 2001, Liu, Sacchetti, Sailhan & Issarny 2005), or provide localised group views (Briesemeister & Hommel 2002) or do not provide virtually synchronous communication (Dolev,

Schiller & Welch 2006). TransMAN provides participating nodes in a MANET with consistent membership views with only a small increase in message delivery latencies over the broadcast message delivery latencies.

Virtually Synchronous Communication in MANETs – None of the existing group communication systems for MANETs provides a virtual synchrony. TransMAN provides virtually synchronous communication with only a small increase in message delivery latencies over the broadcast latencies.

This thesis presents the design, implementation and evaluation of TransMAN. The design presents how TransMAN handles the challenges posed by MANETs, the implementation is provided in the Ruby programming language and the evaluation is carried out by running the TransMAN implementation on GNU systems. Results show that TransMAN adds a small overhead in message delivery latency and that the group membership service reaction time is unaffected by node mobility.

### 1.4.1 Issues Not Covered

TransMAN targets a group communication system for MANETs where the number of nodes are in tens and not in hundreds. Providing a group communication system which provides strong reliability guarantees for large scale systems is a challenge currently under consideration. The Spinglass project (Birman, van Renesse & Vogels 2001) started investigating the area of providing reliable communication with thousands of participants. This thesis does not deal with the issue of scaling a group to such high number of participants.

The broadcast protocol used by TransMAN augments the headers of an existing MANET broadcast protocol to provide reliable message delivery and stability determination. The protocols deliver messages in a FIFO order while higher layers stabilise messages in a total order. Support for ordering schemes like causal or other semantically ordered message delivery is not considered.

Support for a node belonging to multiple groups is also not addressed. While TransMAN provides a virtually synchronous group communication, group membership can some-

times change in such a manner that a state transfer between the participants is required. The provisions for application state transfer during group mergers is not addressed in this thesis.

Also, theoretical analysis of the TransMAN protocols to determine the bounds for message latency and group view change is not addressed in this thesis.

## 1.5 Evaluation Strategy

An evaluation of TransMAN is presented using an implementation of the protocols on GNU systems with IEEE 802.11b wireless communication protocol. Networks of up to ten laptops are used to measure the performance of the protocols with various network configurations.

The evaluation measures the overheads in message delivery latencies when virtually synchronous GCS protocol is used. The overhead is measured relative to the broadcast protocol when no membership service is running. Further, the reaction times for the group communication service are studied while adding and removing nodes from an existing group.

The evaluation shows that the group membership protocol incurs only a small overhead on top of message delivery latencies, while providing ordered reliable broadcast, consistent membership views and virtual synchrony in MANETs.

# Chapter 2

# Related Work

This chapter looks at current communication paradigms applied to MANETs. Work on the use of unicast, multicast and broadcast in MANETs is presented, along with the uses of gossip-based communication protocols. Group communication systems for MANETs are discussed, along with group communication systems for fixed networks. A set of evaluation criteria is listed in the beginning of the chapter and all systems covered in this chapter are evaluated against these.

## 2.1 Evaluation Criteria

The challenges in providing a reliable communication system for MANETs are: (1) frequently changing topologies, (2) lack of collision detection in wireless communication protocols, (3) limited battery power for MANETs and (4) transient network connections. These challenges have generated new approaches to provide communication paradigms like unicast, multicast and broadcast communication in MANETs. In this chapter, the goal of providing a group communication system for MANETs motivates the exploration of these communication paradigms. The approaches to implement these paradigms are also explored, and are evaluated using a set of criteria chosen to capture the ability of each approach to support reliable many-to-many communication in MANETs. The evaluation criteria are:

**Reliable Message Delivery** From (Mullender 1989), message delivery is considered reliable if all participating nodes deliver all messages sent by any participating node. This requirement is necessary to build a group communication system with ordered reliable message delivery and consistent group views. Message delivery is considered unreliable if messages are probabilistically delivered, that is, if messages are delivered only to a subset of the participating nodes or if the sender can not determine which nodes have delivered a message.

**Ordered Message Delivery** The system delivers messages in a well known order, either FIFO, or causal, or total. Applications like collaborative work or mobile multiplayer games in MANETs require ordered broadcast to provide replication management, as shown by Schneider (1990).

**Group View** The system provides each node with a list of other nodes participating in a group. Such a list is useful in implementing message reliability and stability.

**Consistent Group View** The group views provided are consistent at all participating nodes. This is required to support replication management for distributed application by providing reliable message delivery and stability determination.

**Partitionable Groups** Groups are allowed to partition into smaller groups and are allowed to merge to form larger groups. Partitionable groups are required to handle the frequent disconnections and re-connections in MANETs caused by node mobility or failures. Support for partitionable groups creates the next requirement that eliminates uncertainty about message delivery during view changes.

**Virtual Synchrony** There is no uncertainty about messages that are delivered during view changes. Virtually synchronous (VS) communication (Birman & Joseph 1987a) requires that nodes that survive a set of group view changes deliver the same set of messages. VS is important in MANETs where a high rate of group view changes requires that nodes are provided with a reliable knowledge that details which nodes have delivered which messages in which group view.

**Non-Blocking View Changes** Applications are not blocked while a group view change is taking place. The expected high rate of group view changes in MANETs requires that applications are not blocked from sending messages during these group changes otherwise applications will be blocked whenever a node transiently connects or disconnects from the network.

**Topology Independence** The system uses protocols that do not depend on maintaining network topology information. This criterion is motivated by the need to reduce traffic in a MANET, while providing a reliable communication facility. Topology information is often used to develop efficient broadcast protocols for MANETs, but maintaining the topology information in turn can require additional message exchanges for providing location service. Protocols that are independent of location information eliminate the traffic generated by location dissemination or topology maintenance protocols.

**Routing Protocol Independence** The system is independent of information available from routing protocols, for example, routing information. Similar to the argument against the use of topology information, the use of routing protocol generates extraneous message transmissions that are better avoided.

**Redundant Broadcast Reduction** The system is designed to reduce the number of broadcasts required to deliver a message. Apart from avoiding extraneous traffic generated by topology or routing protocols, the system minimises the number of transmissions required to broadcast messages and manage group views.

## 2.2 Communication Paradigms in MANETs

This section looks at the various communication paradigms applied to MANETs and considers their possible use for supporting group communication. The communication paradigms considered are unicast, multicast, broadcast and group communication. Prominent systems for each of these paradigms are evaluated against the evaluation criteria and

results are summarised in Table 2.1, Table 2.2 on page 26, Table 2.3 on page 27 and Table 2.4 on page 33.

| System Type | System | Reliable Message Delivery | Ordered Message Delivery | Group View | Consistent Group View | Partitionable Groups | Virtually Synchronous | Non-Blocking View Changes | Topology Independence | Routing Protocol Independence | Redundant Broadcast Reduction |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Unicasts | UDP (DSR, AODV, ZRP) | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ⊙ | ● | ○ |
| | TCP (DSR, AODV, ZRP) | ● | ● | ○ | ○ | ○ | ○ | ○ | ⊙ | ● | ○ |
| Multicast | MAODV | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| | ODMRP | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ○ |
| | Using Mobility Prediction | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ○ |
| | Team-oriented | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| | DDM | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ○ |
| | LAM | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ○ | ○ |
| | CAMP | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ○ |
| | RDG | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ○ | ● |
| | Anon. Gossip | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ○ | ● |
| | RMA | ● | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ○ |
| | AMroute | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| | BCAST | ● | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● |
| ○ ⇒ No/Not available. ⊙ ⇒ Implementation Dependent. ● ⇒ Yes/Available. | | | | | | | | | | | |

Table 2.1: Unicast and Multicast Protocols in MANETs

### 2.2.1 Unicast

Unicast is the transmission of messages between a pair of nodes. The transmission of messages is either one way or duplex when both nodes send and receive messages to and from each other. In IEEE 802.11b, unicast transmission of messages in a node's neighbourhood is guaranteed to be reliable by using acknowledgements for each unicast message. These

unicasts to neighbours are then carried out in sequence to "push" a message towards a node more than one-hop away. These sequence of unicasts are routed through the MANET so that messages are transported from one end of the MANET to the other while meeting various criteria like: (1) using the shortest route to a destination, or (2) minimise the overall battery consumption at the participating nodes or (3) traverse the links that are least susceptible to failure.

A number of protocols to route messages between nodes have been developed for MANETs, the most widely deployed ones are AODV (Perkins et al. 2003), DSR (Perkins 2001), OLSR (Jacquet, Muhlethaler, Qayyum, Laouiti, Viennot & Clausen 2003), DART (Eriksson, Faloutsos & Krishnamurthy 2004) and ZRP (Haas & Pearlman 1999). These routing protocols determine routes between nodes that are then used to exchange Internet Protocol or IP (Peterson & Davie 2003, Chapter 4) messages. Each message follows the route determined by the routing protocol and this route can change during a sequence of message exchanges. Routes in MANETs change because of the topology changes caused by node mobility, or node failures or the presence of new nodes. The IP messages can be sent using either UDP or TCP protocol which provide non-reliable message delivery or reliable message delivery.

Table 2.1 on the previous page shows that the use of unicast routing protocols fares very poorly against the evaluation criteria of providing a group communication system. The problem with using unicast for providing many to many reliable communication is that multiple copies of messages are sent to accomplish a multicast to more than one receiver.

### 2.2.2 Multicast

To avoid sending multiple copies of a message to reach multiple recipients, one copy is sent along the common route that reaches more than one receivers. Copies of these messages are made when messages have to reach a node that is not on the common route. This approach reduces the number of transmissions required to deliver a message to more than one receivers. IP multicast (Peterson & Davie 2003, Chapter 4) has become a standard for providing multicast to a group of nodes. In IP multicast, a group is identified by a

multicast destination address, and support dynamic membership, i.e., nodes can join and leave the group. However IP multicast does not require reliable message delivery. On fixed networks, IP multicast is handled by multicast routers that follow the Internet Group Management Protocol (IGMP) (Deering 1989). IGMP specifies how participating nodes and gateways interact to provide a multicast service, but it does not specify how multicast is handled by the local area networks (LANs), leaving the LANs to provide multicast in whatever manner that best suits their network characteristics.

Numerous protocols have been developed to provide multicast in a MANET while handling the challenges posed by MANETs. Even if MANETs are treated like LANs, the multicast protocol in MANETs are significantly different from the multicast protocols for LANs. LANs are able to be take advantage of their broadcast domain and multicast is reduced to a broadcast with filters on receiver addresses. Multicast over Wide Area Networks (WANs), which are composed of a number of LANs connected via gateways, is achieved by using multicast gateways between the connected LANs (Peterson & Davie 2003). These techniques can not be directly adopted in MANETs because the MANET broadcast domain is highly unreliable as compared to that of a fixed LAN. Further, the ad hoc nature of MANETs precludes use of designated gateways in MANETs.

The mulitcast protocols for MANETs fall in three categories, depending on how multicast messages are disseminated in the network:

1. Use of a protocol to maintain a structure like a tree, a mesh, or a cluster head. Example protocols include MAODV (Royer & Perkins 1999), ODMRP (Lee et al. 2000, Lee, Su & Gerla 2001), Team-Oriented Multicast (Yi, Gerla & Obraczka 2003), Lightweight Adaptive Multicast (LAM) (Ji & Corson 1998), Core Assisted Mesh Protocol (CAMP) (Garcia-Luna-Aceves & Madruga 1999), AMRoute (Bommaiah, Liu, McAuley & Talpade 1998), and Reliable Multicast Algorithm (RMA) (Gopalswamy, Singhal, Panda & Sadayappan 2002).

2. Message gossips by sending unicast messages to randomly chosen nodes. For example, Route Driven Gossip (RDG) (Jo et al. 2003) and anonymous gossip (Chandra, Ramasubramanian & Birman 2001).

3. Use of optimised broadcasts to provide multicast in an approach similar to the one taken in LANs. Example protocols include DDM (Ji & Corson 2001) and BCAST(Kunz 2003).

Table 2.1 on page 15 shows how each of these protocols fare against the evaluation criteria. The biggest limitations of these protocols is the lack of support for reliable and ordered message delivery, along with a lack of group view management service. As a special case, RMA does provide reliable multicast but does not try to reduce the number of transmissions in the MANET.

One-to-many communication is useful in many situations, where one sender node has to send messages to a number number of nodes that do not need to communicate with the sender. In contrast, a group communication system supports an all-to-all communication allowing all participating nodes to communicate with all other nodes. Supporting such communication in MANETS by extending multicast protocols requires running a number of one-to-many instances, resulting in high message duplication and therefore high battery consumption.

### 2.2.3 Broadcast

The lack of an all-to-all reliable communication and membership service in multicast approaches motivates a look at broadcast protocols currently available for MANETs. Because of the usefulness of broadcast protocols, a number of protocols have been developed for MANETs that handle the challenge of reducing redundant broadcasts. The motivations that have influenced a number of broadcast protocols are: (1) reducing the redundant broadcasts, (2) improving reliability of broadcasts, and (3) determining theoretical minimums for MANET broadcast. The following sections present the prominent protocols grouped by their primary motivations.

#### Reducing Redundant Broadcasts

Broadcast protocols for MANETs that reduce the number of transmissions to propagate a broadcast form the largest group. A reduction in redundant transmissions is useful as less

transmissions results in longer battery life for mobile devices and avoids broadcast storms (Tseng, Ni, Chen & Sheu 2002, Tseng, Ni, & Shih 2001). A broadcast storm results when all nodes rebroadcast all packets that they received from their neighbours. These multiple broadcasts of the same packets lead to a "storm" of packets that congests the network and wastes the limited batteries on mobile devices.

The various broadcast protocols that aim to reduce redundant broadcasts in a MANET are categorised into:

Unstructured approaches – These approaches are independent of any network information. Tseng et al. (2002) propose a number such approaches aimed at handling the broadcast storm problem. Sasson, Cavin & Schiper (2003) propose using probabilistic forwarding of messages to reduce redundant broadcasts.

Structured approaches – These approaches track extra information like node neighbours, node density, node speed, and node location to reduce the number of transmissions in a network (Sun, Feng & Lai 2001, Qayyum, Viennot & Laouiti 2002, Wu & Dai 2003, Agarwal, Cho, Gao & Wu 2004, Lou & Wu 2002, Dai & Wu 2003, Dai & Wu 2004, Wu & Li 1999, Foroozan & Tepe 2005, Wu & Dai 2005, Sucec & Marsic 2000, Wu & Dai 2004). Apart from the unstructured approaches, Tseng et al. (2002) also propose schemes that do use information about the network and these are covered in this category.

The protocols listed for each of the category are not exhaustive but are the most mature and are good representatives of the approaches.

**Using Probabilty**

Using a probabilistic broadcast, each node that receives a message probabilistically forwards the message. This prevents the forwarding of messages by all receiving nodes, which avoids the broadcast storm problem as suggested by Tseng et al. (2002) and Tseng et al. (2001). Potential improvements in broadcast efficiency offered by probabilistic broadcast were studied by Sasson et al. (2003) by verifying the applicability of "phase transition" from percolation theory (Sahimi 1994) to MANETs. They found that phase transition is not applicable to MANET, but probabilistic flooding does improve the rate of successful

delivery of message broadcasts.

**Using Counters**

Like probabilistic broadcasts, the counter-based scheme (Tseng et al. 2002) does not use the knowledge of a node's neighbourhood or the topology of the network to take decisions about forwarding a message or not. The counter-based scheme simply keeps a count of how often a message has been received by a node. If it has been received more than a pre-specified number of counts within a predetermined period of time, the receiving node does not forward the message. Tseng et al. (2002) showed that even a simple counter-based scheme can eliminate many redundant broadcasts, but more informed schemes, like those using location information, eliminate even more of these redundant broadcasts.

Tseng et al. (2002) also suggest schemes that gather information from their neighbours and approach the problem of reducing redundant broadcasts in a structured manner. The structured schemes proposed by Tseng et al. (2002) are: (1) Distance-based scheme, where a receiving node retransmits a message only if the distance to all neighbours that have transmitted the message is smaller than a threshold. (2) Location-based scheme, using the location of nodes to determine if a transmission will cover an additional area greater than a threshold. A simple way to determine if a retransmission will cover additional areas over a threshold was proposed, where a node retransmits a message if it is outside the convex hull formed by nodes from which it has already received the message. (3) Cluster-based scheme, where neighbouring nodes form a cluster of nodes (Lin & Gerla 1997) and the nodes are either normal members of the cluster, or cluster heads or gateways between clusters. The proposed scheme requires that only the cluster heads and gateways forward a message.

Apart from the schemes suggested by Tseng et al. (2002), a number of other schemes have been studied to reduce redundant broadcasts in MANETs (Sun et al. 2001, Qayyum et al. 2002, Wu & Dai 2003, Agarwal et al. 2004, Lou & Wu 2002, Dai & Wu 2003, Dai & Wu 2004, Wu & Li 1999, Foroozan & Tepe 2005, Wu & Dai 2005, Sucec & Marsic 2000, Wu & Dai 2004). All these schemes take the approach of carefully selecting nodes that forward a message and buffer message at receiving nodes while they wait for any duplicate transmissions of the message. These schemes vary on how the nodes that retransmit the

message are chosen and how long these nodes buffer messages while waiting for duplicate transmissions, as follows:

**Using Location**

As an extension of the location-based approach taken by Tseng et al. (2002), Sun et al. (2001) use the location of nodes to determine distances and angles between the nodes. The angle between nodes that have transmitted the message and nodes that have not yet done so is used to determine whether the message should be forwarded or not. A receiving node forwards a message only if it does not receive copies of the message from its neighbours such that the neighbours are spread over all of the 360 degrees around it.

**SBA**

Another simple scheme to reduce redundant broadcasts was proposed by Peng & Lu (2000) as a Scalable Broadcast Algorithm (SBA). SBA uses the idea of maintaining the "cover set" of a message at each node, which includes all the nodes that a node knows have received the message. When a message is first received, the original sender and its neighbours are added to the cover set. While the message is buffered and duplicates of this message are received from other nodes, the other nodes and their neighbours are added to the cover set. If after the end of the buffer period there are some neighbours of this node that are not in the cover set, then the node transmits the message.

**MPR**

Qayyum et al. (2002) propose selecting a multi-point relay (MPR) to propagate link state messages in an optimised OLSR (Jacquet et al. 2003) implementation (OLSR is discussed earlier in Section 2.2.1). The multi-point relays of a node are found by carefully selecting only those one-hop neighbours of a node that can communicate with all of its two-hop neighbours. Qayyum et al. (2002) propose a heuristic for determining a node's MPR. A heuristic is required as they show that the problem of determining MPRs is a NP-complete problem. The heuristic is: (1) starting from an empty set add all one-hop neighbours to the set which are the only neighbours of some one-hop neighbours, (2) for all other nodes in two-hop neighbours add nodes that are one-hop neighbours of this node and that are the neighbours of maximum number of two-hop neighbours. Only nodes in the MPR forward any received messages.

The process of identifying neighbours that should not forward a message so that the least number of nodes forward a message can be looked at in another way. Instead of

finding nodes that should forward a message, protocols determine which nodes should not forward a message. In literature this process of eliminating nodes that should not forward a message has been termed "pruning". A number of protocols have been developed that use the approach of pruning neighbours that should forward the message. Stojmenovic, Seddigh & Zunic (2002) propose a broadcast protocol that construct localised dominant sets using location information and the degree of neighbouring nodes[1]. Some more nodes are pruned from the dominant set to further reduce the number of retransmissions of a message. Each node decides on its own if it should be pruned from the dominant set by using the location information of its neighbours and any duplicate messages it receives.

**Pruning**

Extending the work presented by Stojmenovic et al. (2002), a number of protocols have been developed that determine if a node is in a dominant set or not. These protocols use two to three hop neighbourhood information to determine a local dominant set and then prune nodes from the dominant set, and are listed next.

**Dominating Sets**

1. Using connected dominating sets (CDS) to determine a set of gateways that forward messages (Wu & Li 1999). Some nodes that form the CDS but do not provide additional broadcast coverage are pruned from the CDS. This algorithm was later extended by Dai & Wu (2003) to provide a more general set of rules for pruning some of the gateways from the CDS.

2. Lim & Kim (2001) describe two protocols that prune forwarding nodes. The first one of these protocols uses a simple self-pruning and the second one uses a two-hop neighbourhood information and a dominant pruning (DP) approach to compute the forward node set for each node.

3. Lou & Wu (2002) extend Lim & Kim's (2001) dominant pruning algorithm, reducing the number of two-hop neighbours to be covered by one-hop neighbours. Two algorithms, Total Dominant Pruning (TDP) and Partial Dominant Pruning (PDP) are proposed which use varying degrees of pruning reducing the number of nodes that forward a message.

[1] In graph theory, a dominating set for a graph $G = (V, E)$ is a subset $V'$ of $V$ such that every vertex not in $V'$ is joined to at least one member of $V'$ by some edge.

**Energy Considerations**

Other approaches have looked at energy efficiency of broadcasts (Agarwal et al. 2004, Li, Song & Wang 2005, Liang 2002, Chen, Jamieson, Balakrishnan & Morris 2001, Juan-Carlos Cano 2001). Span (Chen et al. 2001) uses three-hop node neighbourhood information to determine when a node starts forwarding messages by taking on a role of a coordinator. A node bases its decision to become a coordinator on the number of nodes that will benefit from messages forwarded by it, and also on how much battery life it has remaining. The coordinators together form a backbone in the MANET and forward messages based on how much battery life they have. Once a coordinator's battery falls below a minimum, it stops being a coordinator and switches off its radio.

Reducing redundant broadcasts in MANETs has matured with both unstructured and structured approaches providing much improvement over flooding and avoiding broadcast storms. Williams & Camp (2002) provide a good comparison of some of the broadcast techniques mentioned here and show that structured approaches that use location and neighbourhood information perform better than unstructured approaches, especially in dense MANETs.

The broadcast protocols discussed in this section are evaluated against the criteria in Table 2.2 on page 26. The evaluation shows these protocols share the common traits of being topology independent and are routing protocol independent. The reliable broadcast presented in this thesis is can use any of these existing broadcast optimisations.

### Improving reliability of broadcasts

While a lot of work has gone in to developing broadcasts for MANETs that reduce the number of broadcast transmissions, reliable broadcast protocols for MANETs have remained relatively unexplored. The reliable broadcast protocols discussed here are Pagani & Rossi (1997), Pump Slowly Fetch Quickly (PSFQ) (Wan & Campbell 2002), Autograph protocol (Vollset & Ezhilchelvan 2005), and Double-Covered Broadcast (DCB) (Lou & Wu 2004). While these systems improve the reliability of broadcasts they do not provide message stability determination, an important functionality that allows sending nodes to determine that all nodes have received a message. The protocols considered in this section

can not provide these guarantees because they do not track the membership of the network. TransMAN in contrast is able to provide message stability determination because of the membership service provided, which helps determine message stability by gathering implicit acknowledgements for message delivery.

**Using Clusters**

Pagani & Rossi (1997) present a protocol to implement a reliable broadcast for mobile multihop packet networks that uses a clustering protocol. A clustering protocol determines cluster heads in a MANET, such that if two cluster heads have a route between them then the route has at least two or more hops. The nodes on the route between two cluster heads are called gateways and all book-keeping is done at cluster heads. Messages are sent over a "forwarding tree" composed of cluster heads and gateways, and messages are acknowledged by sending acks along the forwarding tree. Gateways and cluster heads collect acknowledgements and send these to their upstream gateways or cluster heads. The challenge is to maintain the cluster in the face of node mobility and also to track acknowledgements when gateways between cluster heads move. Pagani & Rossi (1997) argue that routing protocols should facilitate the management of this information while Stojmenovic et al. (2002) show that cluster based schemes perform poorly in MANETs.

**PSFQ**

Pump Slowly Fetch Quickly or PSFQ (Wan & Campbell 2002), implements reliable broadcast for sensor networks by: (1) localising error recovery and (2) in-sequence forwarding of messages. Localised error recovery means that when a node realises it has missed a message, it sends a nack only to its neighbours. In-sequence forwarding of messages implies that messages received by a node are forwarded in sender FIFO order. Such forwarding guarantees that if a node receives a message, one of its neighbours will have a copy of the message. The broadcast protocol presented in this thesis makes use of both these techniques and avoids two interesting limitations of PSFQ: (1) Because PSFQ uses localised negative acknowledgements, the sender of a message is unable to determine when the message is delivered at all the targeted receivers. This requires receivers to send aggregated delivery reports to the sender. In contrast, the broadcast protocol presented in this thesis does not require sending explicit reports to the source of the message, instead relationships between messages are used to determine the delivery status of messages.

(2) PSFQ uses localised negative acknowledgements under the assumption that the network is composed of static sensor nodes. Such an assumption is not valid in MANETs, and TransMAN's reliable broadcast provides for localised error recovery even in the presence of node mobility.

**Autograph**

The Autograph protocol (Vollset & Ezhilchelvan 2005) introduces the notion of "realising" a message. A message is said to be realised when $(n - f)$ nodes in the network have received the message, where $n$ is the network size and $f$ is the number of nodes that are allowed to fail. The protocol assumes the size of the network is known at the outset and uses bitmaps in messages to track message realisation in a distributed manner. Two more protocols are introduced that use control messages to track message realisation. Vollset & Ezhilchelvan (2005) also debate when the broadcast of a message should be terminated. This is required by Autograph because it does not determine message realisation at all the participating nodes, which is otherwise needed to decide when to terminate a message broadcast. The inability of Autograph to track reliable message delivery or message realisation at all nodes precludes its use for supporting group communication in MANETs that require all to all broadcasts.

**DCB**

Double-Covered Broadcast (DCB) (Lou & Wu 2004) extends the dominating sets approach used by Dai & Wu (2003) to avoid all receiving nodes sending an acknowledgement back to the sender. DCB avoids the transmissions of these acknowledgements by following a scheme of "overhearing message retransmissions". However, DCB does not track membership views and therefore fares poorly against the evaluation criteria. The technique of "overhearing message retransmissions" is also used by TransMAN to reduce the number of message transmissions in a network.

**Sensor Nets**

Finally, broadcasts protocols have also been studied for sensor networks (Orecchia, Panconesi, Petrioli & Vitaletti 2004, He, Stankovic, Lu & Abdelzaher 2003, Wan & Campbell 2002, Ye, Luo, Cheng, Lu & Zhang 2002) where networks of sensors and sinks communicate over a wireless medium. These protocols share a number of challenges from the MANET domain but since sensors and sinks are not mobile, they avoid the challenges of node mobility. Therefore the solutions proposed for sensor networks do not apply directly

to MANETs and these systems are not evaluated here.

Table 2.2 shows how each of the broadcast protocols discussed above fares against the evaluation criteria. These protocols perform the critical function of reducing redundant broadcasts and provide reliable message delivery but do not support a membership service. The reliable broadcast protocol used by TransMAN allows the use of any these protocols and provides efficient reliable broadcast on top of these. TransMAN's flexibility in allowing existing broadcast optimisations supports the use of any future developments in broadcast optimisations in MANETs.

| System Type | System | Reliable Message Delivery | Ordered Message Delivery | Group View | Consistent Group View | Partitionable Groups | Virtually Synchronous | Non-Blocking View Changes | Topology Independence | Routing Protocol Independence | Redundant Broadcast Reduction |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Broadcasts | Unstructured | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● |
| | Structured | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ⊙ | ● |
| Reliable Broadcasts | (Pagani & Rossi 1997) | ● | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| | PSFQ | ● | ○ | ○ | ○ | ○ | ○ | ○ | ● | ● | ● |
| | DCB | ● | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● |
| | Autograph | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● | ○ | ● |
| ○ ⇒ No/Not available. ⊙ ⇒ Implementation Dependent. ● ⇒ Yes/Available. | | | | | | | | | | | |

Table 2.2: Redundant Broadcast Reduction and Reliability in MANETs

### Determining Theoretical Bounds for MANET Broadcasts

Theoretical results have been published that show the bounds for broadcast protocols in both deterministic (Chlebus, Gasieniec, Gibbons, Pelc & Rytter 2000, Gaber & Mansour 1995, Dessmark & Pelc 2001, Uchida, Chen & Wada 2004) and gossip based (Chlebus, Gasieniec, Lingas & Pagourtzis 2001, Gasieniec & Lingas 2002) approaches. These theo-

retical analyses support an understanding of the performance of protocols for MANETs. However, the protocols considered for these analyses assume either synchronous MANETs or the availability of collision detection in wireless networks. These assumptions can not be provided in wireless communication protocols available for MANETs and thus the current theoretical analyses can not be applied to protocols for MANETs.

### 2.2.4 Group Communication in MANETs

| System Type | System | Reliable Message Delivery | Ordered Message Delivery | Group View | Consistent Group View | Partitionable Groups | Virtually Synchronous | Non-Blocking View Changes | Topology Independence | Routing Protocol Independence | Redundant Broadcast Reduction |
|---|---|---|---|---|---|---|---|---|---|---|---|
| GCS (MANETs) | (Roman et al. 2001) | ○ | ○ | ● | ● | ○ | ○ | ○ | ○ | ● | ○ |
| | (Meier et al. 2001) | ○ | ○ | ● | ○ | ● | ○ | ○ | ○ | ● | ○ |
| | (Dolev et al. 2006) | ● | ● | ● | ● | ● | ○ | ● | ○ | ● | ○ |
| | (Bar-Yossef et al. 2006) | ○ | ○ | ● | ○ | ○ | ○ | ○ | ● | ● | ○ |
| | (Friedman 2003) | ○ | ○ | ● | ○ | ○ | ○ | ⊙ | ⊙ | ⊙ | ⊙ |
| | (Dolev, Gilbert, Lynch, Shvartsman & Welch 2003) | ● | ● | ● | ● | ○ | ○ | ● | ● | ○ | ○ |
| | (Briesemeister & Hommel 2002) | ○ | ○ | ● | ○ | ○ | ○ | ○ | ● | ● | ○ |
| | (Liu et al. 2005) | ○ | ○ | ● | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| ○ ⇒ No/Not available. ⊙ ⇒ Implementation Dependent. ● ⇒ Yes/Available. | | | | | | | | | | | |

Table 2.3: Group Communication Systems For MANETs

Reliable communication protocols and algorithms that reduce redundant broadcasts are critical for application development in MANETs, but membership view and ordered message delivery provide application developers with guarantees that ease the task of programming distributed applications (Birman & Joseph 1987a). The programming task

is made easier because of the guarantees provided. This section looks at the current state of the art in group communication for MANETs. All the group communication systems covered in this section are designed to handle the challenges posed by MANETs, like frequent topology changes and the limited battery lives of mobile nodes. However, to address the challenges, current systems either sacrifice strict reliability guarantees or make assumptions about MANETs that are hard to satisfy, as discussed next.

**Using Location**

Roman et al. (2001) propose the use of information about the geographical locations of nodes to maintain consistent group views, while assuming reliable FIFO communication channels between mobile nodes. Each node advertises its location to a group leader, which serialises all group view changes. The protocol assumes the availability of such a group leader and does not address the need for a leadership election algorithm. The group leader makes decisions on the membership of the group and communicates it to all nodes in the group. To take decisions on the membership of the group, the leader uses the relative location of nodes and assumes the maximum possible velocity to predict network partitions. Once the leader predicts a network partition, it informs all other nodes of the change in the network and these nodes then initiate a view change. The view change protocol requires that nodes stop sending any further messages, flush all the unsent messages to other nodes in the group, and finally change the group view once they have received all the flush messages from all other group members. The approach provides consistent group membership, but does not provide reliable broadcast or virtual synchrony. The protocol also does not consider any approaches for reducing the number of transmissions in the network.

**Multiple Coordinators**

Pradhan & Helal (2003) describe a protocol for providing serialised view formations. Like Roman et al. (2001), the protocol uses a coordinator-based approach by extending the agreement protocol by Abbadi, Skeen & Cristian (1985). A node that detects a potential change in group view initiates a group view change protocol and acts as a coordinator for view change agreement. A number of such view changes can be initiated by different coordinators and the protocol guarantees serialised view changes by assigning a unique group view identifier to each group view change initiation. However, the following limitations

result in the protocol faring poorly against the evaluation criteria. (1) The protocol does not state when a view change is initiated and requires an application to independently decide to trigger the view change protocol. (2) The protocol stops application broadcasts while a group view change is taking place, resulting in a large number of blocking periods because of the high number of expected group view changes in MANETs. (3) The protocol assumes that reliable communication channels are available and does not address provision of either reliable broadcast or virtual synchrony.

**Proximity Groups**

Meier et al. (2001) present the concept of "proximity groups" where group membership depends on the location of mobile nodes. Each group has an associated geographical area, called proximity, and nodes can join the group only if they are inside the proximity. The proximity of a group can either be a fixed geographical area or can be relative to a mobile node, allowing for mobile proximities that move when the associated node moves. Meier et al. (2001) also present an approach to determine the area covered by the group at any time and propose a partition anticipation scheme based on information about the area covered by the group. While being concerned about group communication, the work does not propose a solution for a membership service or reliable delivery of messages. The lack of these services results in the work faring poorly against the evaluation criteria.

**Fuzzy Views**

Friedman (2003) describe a fuzzy membership approach for tackling MANET's challenges of frequently changing topology and highly unreliable communication links. Fuzzy membership provides each participant with a list of nodes that have a fuzziness level associated with them. The fuzziness level for each node in the membership list indicates how reliable is the connection to that node. Each node uses a fuzzy failure detector (Friedman & Tcharny 2003a, Friedman & Tcharny 2003b) that maintains information about all other nodes as being in one of the three states – suspected, fuzzy suspected or not suspected. The information from fuzzy failure detector is used to implement the fuzzy membership specification, which is then used to determine when a message has been delivered at all nodes in the fuzzy group view. Friedman (2003) also introduce the notion of fuzzy stability, where a message is considered fuzzy stable if it has been received by all nodes that are neither suspected, nor fuzzy suspected. The algorithms and implementation for fuzzy group

membership are not published yet and only the specifications are available. To support implementations of fuzzy membership specifications, TransMAN introduces the notion of "transient views" (Section 4.5.2), which can provide a view on nodes that can be considered fuzzy members of the group.

**Geo-Quorums**

The GeoQuorum system by Dolev et al. (2003) approaches the problem of providing reliable communication in MANETs by conceiving a read/write shared memory implementation. The key idea is to associate abstract atomic objects, called focal points, with geographical areas. The focal points provide atomic read/write operations on a virtual shared object using the GeoQuorums algorithm. Nodes that populate a focal point are used to implement a totally ordered one-hop broadcast while assuming a global time service like GPS. These focal points are then connected by a geo-cast service (Ko & Vaidya 1998) that maintains read and write quorums of focal points, providing an implementation of share memory across focal points. The problems with this approach are the dependence on a time and location service and the need for a well known list of focal points along with their mappings to geographical locations. Since GeoQuorums do not claim to provide a group communication, they do not try to provide a group view or virtual synchrony. However, because the system provides a shared memory service, it could be extended to implement a group communication service.

Other approaches taken by systems providing group communication for MANETs are based on the use of (1) random walk (Barber & Ninham 1970), (2) probabilistic broadcast (Sasson et al. 2003, Eugster, Guerraoui, Handurukande, Kouznetsov & Kermarrec 2003) or (3) gossip (Birman, Hayden, Ozkasap, Xiao, Budiu & Minsky 1999). A random walk is a formalisation of the intuitive idea of taking successive steps, each in a random direction. Probabilistic broadcasting (also looked at in Section 2.2.3 on page 18) is the idea of each node broadcasting a message with a certain probability. Gossip techniques (also covered in Section 2.2.2) use the idea of communicating with a number of randomly chosen nodes. The systems that use these techniques are presented next along with the problems of using these approaches to provide a group communication system.

**Agent Random Walk**

Dolev et al. (2006) use the random walk of an agent to provide a self stabilising (Dolev

2000) probabilistic group communication system for MANETs that does not require a coordinator, or a distributed structure like a routing path or a tree. The probabilistic nature of message reliability and group views requires the interpretation of these guarantees by the application developer. Further, the use of a mobile agent to implement a total order broadcast service results in high message latencies, caused by the $O(n^3)$ agent moves required before a message is delivered to all nodes. Requiring such a high number of reliable agent moves, when the communication links in MANETs are highly unreliable, further increases the message latencies. Even the group membership service requires $\Omega(n^3)$ agent moves before a view change can take place. Such high latencies and probabilistic guarantees discourage the use of random walk approach for implementing a GCS in MANETs.

**RawMS**

Random walk has been used by Bar-Yossef et al. (2006) to implement RawMS, a membership service that provides each participating node with a randomly uniform membership view. In contrast to the work by Dolev et al. (2006), RawMS does not require an agent to visit each node, but instead messages are dispersed in a MANET using a random walk. This avoids the high latencies associated with one agent visiting all the nodes in a MANET. In RawMS, each node sends a message equivalent of an "I am alive" message using a random walk and the node where the random walk terminates updates its membership view to reflect the reception of the "I am alive" message. The protocol does not depend on a routing protocol or any topology information. However, RawMS fares poorly against the evaluation criteria because of the lack of strong guarantees like consistent membership view or ordered message broadcasts.

**PILOT**

PILOT (Luo et al. 2004) uses the principles of gossipping to disseminate messages. In PILOT, nodes use unicast to gossip messages to random nodes in the network, selected using information available from a routing protocol. Gossip-based broadcast is made reliable by using negative acknowledgements to gather lost messages. Gossip-based broadcast is also used to provide each participating node with a random partial view of the group, which is then used to provide a data sharing service across the MANET. Just like other probabilistic systems, PILOT fares poorly against the evaluation criteria because of no support for consistent membership service, ordered message delivery or virtual synchrony.

**Neighbourhood Groups**

Briesemeister & Hommel (2002) presents a "localised group communication service" for MANETs, which uses periodic beacons to maintain the membership status of each node's one-hop neighbours. The membership service provides timeliness guarantees for view change latencies. Since the membership service is localised, it is not concerned with consistent group views or view partitions and mergers. These limitations result in the localised group communication service faring poorly against the evaluation criteria.

Liu et al. (2005) present a classification of group membership services according to network model, location, scale, trust, or QoS awareness. Followed by the classification, they propose a group membership service that assumes a routing protocol and uses a leader to manage group membership and inter-group communication. There are two problems with the proposed solution – (1) the use of leadership election in dynamic environments like MANETs and (2) the use of a routing protocol. Both these approaches results in an increase in the number of message transmissions in the network. Further, the proposed solution does not address virtual synchrony.

**Sensor Networks**

There is also work that focusses on GCS for sensor networks (Gavidia, Voulgaris & van Steen 2005, Vieira & Rosa 2005, Liu, Liu, Reich, Cheung & Zhao 2004). Sensor networks constitute of energy constraint devices communicating over a wireless medium. The devices or sensors in a sensor network are immobile and therefore the sensor networks do not face the same challenges as MANETs. Therefore, the work in sensor networks is not covered here.

## 2.3 Group Communication in Fixed Networks

Group communication systems are under investigation for a long time with systems like V Kernel (Cheriton & Zwaenepoel 1985), ISIS (Birman & Joseph 1987b), Transis (Dolev & Malki 1995), Totem (Amir et al. 1995), Timewheel (Mishra, Fetzer & Cristian 2002), Amoeba (Kaashoek et al. 1993), Newtop (Ezhilchelvan et al. 1995), RELACS (Babaoğlu, Davoli, Montresor & Segala 1998), Phoenix (Malloth & Schiper 1995, Babaoğlu & Schiper 1995b), Moshe (Keidar et al. 2002), Spinglass (Birman et al. 2001), Jgroup (Meling, Montresor, Babaoglu & Helvik 2002), Ensemble (Birman et al. 2000) and Horus (van Renesse

| System Type | System | Reliable Message Delivery | Ordered Message Delivery | Group View | Consistent Group View | Partitionable Groups | Virtually Synchronous | Non-Blocking View Changes | Topology Independence | Routing Protocol Independence | Redundant Broadcast Reduction |
|---|---|---|---|---|---|---|---|---|---|---|---|
| GCS (Fixed Networks) | V Kernel | • | ○ | ○ | ○ | ○ | ○ | ○ | – | – | – |
| | ISIS | • | • | • | • | ○ | • | ○ | – | – | – |
| | Transis | • | • | • | • | • | • | • | – | – | – |
| | Totem | • | • | • | • | • | • | • | – | – | – |
| | Consul | • | • | • | • | ○ | ○ | • | – | – | – |
| | Timewheel | • | • | • | • | ○ | – | • | – | – | – |
| | Amoeba | • | • | • | • | ○ | ○ | ○ | – | – | – |
| | Newtop | • | • | • | • | • | • | • | – | – | – |
| | RELACS | • | ○ | • | • | • | • | ○ | – | – | – |
| | Phoenix | • | • | • | • | • | • | ○ | – | – | – |
| | Moshe | – | – | • | • | • | • | – | – | – | – |
| | Spinglass | ○ | ○ | • | ○ | • | ○ | • | – | – | – |
| | Jgroup | • | • | • | • | • | • | • | – | – | – |
| | Ensemble | • | • | • | • | • | • | • | – | – | – |
| | Horus | • | • | • | • | • | • | • | – | – | – |

○ ⇒ No/Not available. ⊙ ⇒ Implementation Dependent. • ⇒ Yes/Available.
– ⇒ Not Applicable

Table 2.4: Group Communication System For Fixed Networks (LANs and WANs)

et al. 1996, Friedman & van Renesse 1996). These systems provide reliability guarantees among a group of nodes in LANs or WANs. The network model of LANs and WANs is different from a MANET, in that the nodes in a LAN are connected via highly reliable links, topologies do not change as often as in MANETs and the power supply or battery life of participating nodes is not a concern. The long distance communication links in a WAN are much more stable than a MANET and communication in a WAN is not broadcast based, as in MANETs. These differences in system requirements preclude the use of solutions from fixed networks directly for MANETs, but these systems are discussed here as TransMAN builds on many of the concepts and algorithms used to implement these

systems. The next sections present some of the prominent systems and their approaches for building a GCS for fixed networks. Each of these systems is evaluated against the criteria presented in Section 2.1 on page 12 and the results are shown in Table 2.4.

**V Kernel**

V Kernel (Cheriton & Zwaenepoel 1985) first proposed the abstraction of process groups to provide inter-process communication, along with process and memory management in a distributed operating system. The V Kernel uses process groups to facilitate a client's communication with a group of servers, or to suspend and resume a group of processes working on a connected task or for intra-process communication in a distributed application. V Kernel introduced an API for communicating with a group. Using the V Kernel process group API, a process can send and receive messages to a group, join or leave a group, create or destroy a group and even kill or suspend all members of a group. Since this process group API was just emerging at the time of V Kernel, properties like message ordering, partitionable groups and virtual synchrony were not considered.

**ISIS**

Followed by V Kernel, the ISIS group communication system (Birman & Joseph 1987b, Birman & Joseph 1987a) proposed the idea of providing process groups with a virtually synchronous (VS) inter-process communication. VS communication makes the message broadcasts, group membership changes and activity migration appear synchronous on all nodes, which allows the use of weak message ordering requirements to implement synchronous communication between processes. The idea behind VS communication is to "synchronise" only those events that can affect the state of applications running at the participating nodes. For example FIFO ordering of messages can provide for a replicated FIFO queue at a number of processes without requiring atomic broadcast. VS will guarantee that processes receive the updates to the queue in FIFO order. VS has been widely adopted for implementing group communication services as it eases distributed application development (Birman 1993). A number of models of VS have been proposed since then (Friedman & van Renesse 1996, Moser, Amir, Melliar-Smith & Agarwal 1994) and they are discussed in the context of the systems that propose them.

ISIS uses a "view manager" to coordinate view changes. All group view changes are decided on by this view manager, who communicates its decisions to member nodes using

a 2-phase commit protocol. To handle the situations where the view manager fails during a view change, the view manager stops propagating application messages during a view change. The view manager is allowed to incorrectly decide on the status of a node and the node is then required to put in a request to join the group again. ISIS requires that processes stop participating in the group if the number of nodes in the group falls below a pre-defined count. The use of a view manager and blocking applications during the view changes results in the system's poor rating against the evaluation criteria.

### 2.3.1 Using Broadcast Domains for Group Communication

Systems for providing reliable communication in broadcast domains are considered next. TransMAN extends on a number of concepts from these systems and therefore these systems are evaluated in detail as compared to the earlier systems.

A broadcast domain is composed of a number of communicating nodes that share the same broadcast medium and where a message transmission by one node reaches all others without the use of forwarding or gateways. Ethernet (Metcalfe & Boggs 1976) is an example of a broadcast domain. The nodes at the intersection of broadcast domains act as gateways between them by relaying messages from one broadcast domain to the other. The wireless radio range of a node is another broadcast domain. In this thesis, MANETs are viewed as being composed of more than one intersecting wireless broadcast domains.

Broadcast domains have been studied under fixed network scenarios by Peterson et al. (1989) for tracking message relationships as conversations in Psync and by Moser, Melliar-Smith & Agarwal (1994) for the Trans broadcast protocols. The approach taken by Dolev & Malki (1995) for their Transis group communication system is a development of the ideas proposed in Psync and Trans. These systems are not designed for the system model of mobile ad hoc networks and therefore fail to handle the challenges created by these new dynamic networks. This limitation is visible across all the systems discussed in this section and Table 2.4 on page 33 makes this limitation apparent. However, these approaches for efficiently implementing reliable broadcast communication in a broadcast domain have inspired the approach taken in TransMAN.

**Psync**

The Psync protocol presented by Peterson et al. (1989) first introduced the idea of tracking message ordering by observing messages as "conversations" between nodes. The conversation abstraction provides a *shared message space* through which messages are exchanged between participating nodes. This message space is defined by a directed acyclic graph that preserves partial order between messages. This partial order is then converted to a total order using the properties of the conversation. The Psync broadcast protocol is extended to provide a membership, described in the Consul (Mishra et al. 1993) group communication service which is covered on page 38.

**Trans**

Melliar-Smith & Moser (1993) extended the idea of a conversation and introduced the use of transitivity of message acknowledgements, to provide reliable communication despite a noisy communication medium. Extensions to the Trans broadcast protocols then provide a total order on the messages broadcasted. The total ordering of messages in Trans is achieved without extra message broadcasts before a message can be delivered. These protocols take advantage of the fact that high reliability broadcast domains rarely lose a message and therefore provide a high probability of totally ordering messages requiring only one message transmission. Using the total order delivery of messages Moser, Melliar-Smith & Agarwal (1994) construct a membership service that can handle new nodes joining an existing broadcast and can remove failed nodes from the broadcast without blocking message transmissions. These membership service protocols however do not deal with virtually synchronous communication. The shared message space concept from Psync remains central to Trans's total ordering and membership service protocols.

There are two main ideas to take away from Psync and Trans: (1) tracking message dependencies in a shared message space (from Psync), and (2) gathering implicit message acknowledgements to provide reliable broadcast and to reduce the number of message transmissions in a broadcast domain (from Trans). These ideas can be applied to the overlapping broadcast domains in MANETs. The work described in this thesis provides a reliable broadcast protocol and a virtual synchronous group membership service using the ideas from Trans and Psync.

**Transis**

Dolev, Malki & Strong (1994) describe Transis, a non-blocking partitionable group

communication service. A membership service that allows participating nodes to leave and join a communication at any time is a "partitionable group communication" service. Transis uses the ideas from Psync and Trans to implement a multicast service that provides a number of message ordering schemes (Dolev, Kramer & Malki 1992). The multicast service is used to provide a group membership service for LAN like broadcast domains. Multicast services from LANs are joined together using specially designated gateways that relay messages from one LAN to the other. Transis therefore works over WANs with arbitrary topologies of connected broadcast domains. TransMAN uses the idea of maintaining groups over a composition of broadcast domains. However, in contrast to the broadcast domains in the Transis system model, the broadcast domains in MANETs are not connected by one designated gateway. Instead, a number of nodes that can simultaneously act as gateways and these can change because of the dynamic topology of MANETs.

While TransMAN is influenced by the Transis approach of combining broadcast domains there are two key differences in the network model: (1) the degree of reliability of the broadcast domains and (2) the use of designated gateways between broadcast domains. TransMAN addresses these differences to provide a partitionable, virtually synchronous group communication for MANETs.

**Totem**

Totem (Amir et al. 1995) uses a total order broadcast to provide a group communication system that provides ordered message delivery with Extended Virtual Synchrony (EVS) (Moser, Amir, Melliar-Smith & Agarwal 1994). In the presence of group partitions, merges, and recovery, EVS guarantees that messages are delivered in the same order across all partitions of a group. EVS uses the idea of transitional membership where: nodes that are expected to be in the next group view are informed of this transitive view; messages are delivered as normal in this transitive view; and a regular group view is installed when agreement is reached on it. The Totem membership agreement protocol uses a total order broadcast that is in turn implemented using a token-ring protocol. The token-ring protocol is executed in broadcast domains, which are then connected to each other via designated gateways. The approach taken in Totem is a mismatch for composite broadcast domains in MANETs. However, the idea of using transitive views is extended in TransMAN and an

agreement protocol is proposed that can provide applications with such transitive views.

Consul (Mishra et al. 1993) uses shared message space and partial order of message delivery developed in Psync (Peterson et al. 1989) to provide a totally and semantically ordered delivery of messages, and a membership service. Semantically ordered delivery of messages exploits the commutativity of operations in certain applications by totally ordering only those messages that have a commutative relationship, reducing the overhead of totally ordering messages. Consul's membership service uses explicit acknowledgements from participating nodes to decide on the next group view. In contrast, TransMAN uses implicit acknowledgements from the shared message space to determine a group view change, reducing the number of transmissions in the network. Consul's membership service does not address group mergers, while TransMAN allows groups to merge and partition by allowing the shared message space to merge and partition. **Consul**

### 2.3.2 Solutions Not Using Broadcast Domains

Other group communication systems do not use broadcast domains to implement its services and are designed to work over WANs, which are networks where broadcast communication is not available. If the principles used in these systems are applied to MANETs, the GCS will fail to take advantage of the broadcast based radio communication protocols available in MANETs, which will result in high network traffic. These systems are covered here because the concepts developed are either used in or provided by TransMAN. For example, these systems develop models for Virtual Synchrony and propose agreement protocols that TransMAN builds on.

The Timewheel group communication system (Mishra et al. 2002) models the network as a timed asynchronous network (Cristian & Fetzer 1999) and provides timeliness guarantees for message delivery and group view changes. Timewheel is a failure aware group communication system, which means that nodes are made aware of other nodes' failures or a failure of the system to meet its timeliness guarantees. To provide timeliness guarantees, Timewheel assumes each node has a hardware clock and provides a clock synchronisation protocol to synchronise them. A leader based approach is used and the role of a leader is **Timewheel**

rotated among the group members. Group membership is agreed upon using a majority agreement protocol. The use of synchronised clocks and the coordinator-based approach make the solution unsuitable for MANETs where high variations in hardware result in asynchronous networks and frequent topology changes can result in frequent re-elections for coordinators.

**Amoeba**

Amoeba is a distributed operating system that provides group communication semantics. Unlike the V Kernel (discussed on page 32), Amoeba's process group semantics (Kaashoek et al. 1993) provide reliable and ordered messages delivery. Being a distributed OS, Amoeba makes use of a node's ability to block other nodes, and uses such blocking to provide node specified resilience of broadcasts: each node sends a message with a resilience parameter and the node returns control to the application only after a resilience number of group members have received the message. Amoeba uses a sequencer (a coordinator) to provide the process group communication and all nodes unicast messages to the sequencer. The sequencer then broadcasts these messages to other group members, returning control to the sender once a "resilience" number of nodes have acknowledged the delivery of that message. Nodes then piggyback acknowledgements when they send other messages. In case of a node failure, Amoeba initiates a two phase protocol that involves electing new leaders and then restarts the group membership protocol. Such an approach that uses coordinators and resets the membership protocol in case of node failures is not suitable for MANETs where high number of node failures and restarts are expected.

**Newtop**

Newtop (Ezhilchelvan et al. 1995) uses logical clocks to implement a group communication system for wide area networks. Newtop allows nodes to belong to multiple groups and supports dynamic formation of new groups. The key idea used is that of causal blocks, where each causal block consists of message that are concurrent; and messages in concurrent blocks do not have a causal relationship between each other. These blocks are then assigned unique block numbers and all messages carry this block number, identifying the block to which they belong. These blocks are totally ordered using a sequencer. Sequencers from overlapping groups cooperate together to provide total ordering across the overlapping groups. The virtual synchrony semantics supported by Newtop are more

flexible than ISIS because, unlike ISIS, Newtop delivers messages across view changes. However, this is at the cost of blocking nodes from sending a message while waiting for responses from the sequencer. The sequencer is also central in the Newtop's membership protocol, which requires all messages to be multicast through the sequencer and therefore requires the sequencer to a-priori know about all the other nodes that want to participate in a group. An a-priori knowledge of nodes that want to join the group and the use of sequencer based agreement protocol makes Newtop highly unsuitable for MANETs where group membership is unknown at the outset and a high rate of changes is expected.

**RELACS**

RELACS (Özalp Babaoğlu, Davoli, Giachini & Baker 1995) is a group communication system designed to support groups spanning large geographical distances. RELACS approaches the problem by treating all participating nodes as connected by point to point communication channels. A failure detector pings other nodes to determine if they are alive or not. Information from the failure detector is used to trigger a view change protocol, which uses a coordinator to decide on the next view. The group view change protocol guarantees that delivery of messages is totally ordered with respect to group view changes. However, RELACS does not provide ordered message delivery leaving it to be provided by higher layers. Further, the dependence on explicitly pinging all participating nodes would result in high traffic in MANETs. TransMAN avoids pinging each node to determine node failures, and instead extrapolates node failures from broadcasts of application messages.

**Phoenix**

Schiper & Ricciardi (1993) describe a virtually synchronous communication system that is used to provide the Phoenix (Malloth & Schiper 1995, Babaoğlu & Schiper 1995b) group communication system for networks spread over larges distances. Phoenix uses the lower layers that provide view synchrony to provide a totally ordered broadcasts. Phoenix structures groups as containing a core set of members, client members and sink members. This group structure is a solution for scaling groups to large areas. However, the solution requires blocking applications from sending messages while a group view change is taking place. The approach of imposing a hierarchical structure on nodes and blocking applications during view changes is unsuitable for the flat structure of MANETs. However, the solution uses a reduction from consensus to view synchrony that is also used by TransMAN.

**Moshe**

Keidar et al. (2002) present a group membership service (Moshe) for WANs by following an architecture where the membership information is maintained only by a set of dedicated servers and not by all participating nodes. Such an architecture is motivated to scale groups to a large number of participants over a WAN. However, the use of such an approach in MANETs would require some nodes to act as designated servers. This would result in a channelling of a large amount of traffic through these designated gateways, which contradicts the flat architecture of MANETs. Besides, channelling traffic through a few nodes would consume unequal energy at the participating nodes, again contradicting the flat architecture of MANETs. On the other hand, Moshe delivers only the current membership views, choosing to abstain from installing any views if the network is unstable. TransMAN follows a similar approach where views are not installed if the network is unstable. This avoids installation of stale views, which is essential in MANETs where a high rate of view changes is expected.

**Spinglass**

The Spinglass project (Birman et al. 2001) is an attempt to provide a suite of reliable communication protocols designed to react efficiently to perturbation in a large group spread over a large area. The project's motivation is to use Bimodal Multicast (Birman et al. 1999) to implement both reliable message dissemination, group membership and virtual synchrony. The term Bimodal implies the protocol runs in two modes: the first where nodes multicast messages without any reliability semantics, and the second when nodes gossip with each other to find missing messages they did not receive and then recover them. SWIM (Das et al. 2002) presents Spinglass's membership service, which separates failure detection from membership information dissemination. The membership uses a gossip-based failure detector (Gupta, Chandra & Goldszmidt 2001) and if a node suspects another node it removes the other nodes from its membership and disseminates the suspicion information to other nodes. SWIM results in constant overhead on membership changes with respect to the number of group members. The Spinglass approach is unsuitable for MANETs because: (1) the scale of systems envisioned for Spinglass and MANETs are different and (2) Spinglass requires unicasts of gossip messages between nodes. However, unicast routing protocols for MANETs of sizes envisioned by Spinglass is still a problem

under study[2].

Jgroup (Meling et al. 2002, Babaoğlu, Davoli & Montresor 1998) provides the notion of a Group Method Invocation (GMI), which is an extension of the Java RMI interface to invoke method calls on remote objects. GMI allows a client to communicate with a server group completely oblivious of the server as a group of processes. Jgroup provides a group membership service along with reliable communication and a state merging service. The Jgroup membership protocol uses an agreement to install consistent views at participating nodes. The agreement protocol is coordinator-based and in case of a coordinator failure the nodes elect a new coordinator. The use of a coordinator to provide agreement contradicts the flat hierarchy of MANETs and is avoided in TransMAN. Instead, TransMAN provides a group view agreement protocol that does not depend on a coordinator and uses application messages to reach agreement on group views.

**Jgroup**

### 2.3.3 Virtual Synchrony

This section presents a background of various virtual synchrony models proposed in literature including the abstraction proposed by Chockler et al. (2001). This abstraction is then used to specify the virtual synchrony model supported by TransMAN.

In a view-oriented group communication service, **send** and **recv** events occur in the context of a group view. Some GCS guarantee that all members deliver a message in the same view as the one in which it was sent, guaranteeing *Sending View Delivery*. A problem with such a requirement is that applications need to block when sending messages while a new view is being installed.

A weaker alternative to *Sending View Delivery* is suggested as *Same View Delivery*, which requires that all members deliver a message in the same view. The view messages are delivered in may or may not be the view the message was sent in. The property suffices for applications which do not care which view the message was sent in, but are satisfied by the knowledge that all members receive the message in the same view. Using *Same view delivery*, applications continue sending messages while group view change is taking place,

[2] http://moment.cs.ucsb.edu/AODV/ID/draft-ietf-manet-aodv-13.txt

the service guarantees messages sent during a change of view are delivered in either of the two views (the original or the changed view). All GCSs satisfy *Same View Delivery* for messages sent/received in the context of some view.

As and alternative to the above two properties, Friedman & van Renesse (1996) suggest *Weak Virtual Synchrony* (WVS), a model stronger than *Same View Delivery* yet weaker than *Sending View Delivery*. It is weak enough for applications to continue sending messages while a view is changing, yet strong enough to provide some notion of support for "view-aware" applications. *Weak Virtual Synchrony* requires each view change to be preceded by a "suggested view", a further requirement placed on every message sent in the suggested view (viz. is a superset of the current view) is that they should be delivered in the next regular view. Thus "view-aware" applications know each message they receive was either sent in the current view, or the preceding suggested view.

Even though the the WVS model does not require applications to block sending messages, two main problems arise. First, WVS does not allow new nodes to join the next regular group once a suggested view has been proposed. If a new node wants to join the group after a suggested view has been proposed, the service initiates a new view install as soon as the next regular view is installed. This causes the system to never stabilise in the face of frequent changes, a condition often expected in MANETs. The second problem with WVS is its focus on applications that are satisfied to know a superset of the existing view, and do not need the precise set of nodes receiving a message.

Sussman, Keidar & Marzullo (2000) proposed the *Optimistic Virtual Synchrony* (OVS) model that requires each view installation to be preceded by an optimistic view event. The optimistic view provides the application with a "guess" about what the next view will be. After this event, applications "optimistically send messages" assuming they will be delivered in a view identical to the optimistic view. If the next view that is installed is not identical to the optimistic view, the application chooses to use either the messages received or roll back the optimistic message. The OVS approach runs into the same problems as the WVS, but has an additional problems that the suggested view is just a guess and applications can use OVS only if they support rolling back of their operations.

Moser, Amir, Melliar-Smith & Agarwal (1994) proposed the *Extended Virtual Synchrony* (EVS) that requires that messages are delivered in the same order across all partitions and mergers of a group. EVS uses the idea of transitional membership where nodes that are expected to be in the next group view are informed of this transitional view. This is in contrast to WVS and OVS where the transitional views are either a superset of the the next group view or are just a plain guess. EVS allows message broadcasts and delivers messages as normal while a transitional view is proposed. A regular group view is installed when nodes agree on it. EVS has the benefit over OVS and WVS that the transitional views are related to the views that precede and follow the transitional views. This benefit has limitation that the GCS supporting EVS has to offer a protocol supporting the management of transitional views.

### 2.3.4 Protocol Stacks For GCS

Apart from these group communication systems, protocol kernels like Ensemble (van Renesse et al. 1996, Birman et al. 2000) and Appia (Miranda, Pinto & Rodrigues 2001) have been developed to allow easy composition of group communication protocols. These protocol kernels allow composition of a group communication system from components or layers that interact with each other using predefined interfaces. These stacks are currently limited to the use of a coordinator, as all the layers expect to interact with a coordinator. All message broadcasts and group view changes go through this coordinator. The coordinator based approach is not suited for MANETs and therefore extensions to these protocol kernels are not considered in this thesis.

**Horus and Ensemble**

The Horus(van Renesse et al. 1996) system was developed to handle the complexity of programming a monolithic system like ISIS. Horus employed a stack of "micro protocols" each of which communicates with one such protocol above it and one below. Ensemble(Birman et al. 2000) was developed to rebuke the skeptics who questioned the correctness of Horus, and allows the use of theorem checkers to verify the implementation of the micro protocols and the complete system.

Jazz Ensemble[3] is an extension of Ensemble that targets the MANET domain. The problem with using Jazz Ensemble for MANETs is the apparent slowdown in communication because of a dependence on a coordinator to order messages and decide on group view changes. Routing messages through a coordinator in a MANET results in increased latencies, especially while facing the highly unreliable communication links and frequent topology changes in MANETs. **Jazz Ensemble**

The work on group communication systems for fixed networks resulted in a formal understanding (Babaoğlu et al. 2001, Chockler et al. 2001, Babaoğlu, Davoli & Montresor 1998) of the properties supported by various group communication systems. These studies of formal specifications help in verifying the guarantees provided by a GCS. The work presented in this thesis is specified using the properties studied in these works.

## 2.4 Conclusion

The chapter considered the different communication paradigms that are available in MANETs and also the existing group communication systems from MANETs and fixed networks. The existing unicast and multicast protocols for MANETs are not useful in implementing an all-to-all communication system. Unicast is unsuitable because it propagates as many copies of a messages in the network as there are recipients. Multicast communication, on the other hand, is targeted to provide efficient communication between mutually exclusive senders and recipients. Such communication does not suit the needs of an all to all communication required to be provided by a group communication service.

The broadcast protocols for MANETs focus on either reducing redundant broadcasts or providing reliability. Both these motivations are important for an efficient implementation of a group communication system in MANETs. TransMAN provides a means to implement reliable broadcast while using any of the existing techniques to reduce redundant broadcasts in the network.

The existing group communication systems for MANETs either use optimisation techniques that result in probabilistic solutions for group membership and broadcast, or make

[3] http://www.cs.technion.ac.il/Labs/dsl/projects/JazzEnsemble/index.html

assumptions that are hard to provide in MANETs. There is a need for a group communication systems in MANETs that can provide deterministic reliable broadcast along with a partitionable group membership service that handles the challenges presented by the dynamic nature of MANETs.

Finally, the chapter looked at group communication systems from traditional wired networks. The systems taking advantage of the broadcast communication in LANs have inspired the solution proposed in this thesis. Other system for WANs, motivate the use of models like extended virtual synchrony, which suit the dynamic nature of MANETs.

# Chapter 3

# Model

This chapter presents an Application Programming Interface (API) for TransMAN and assumptions made for designing the system. This is followed by the system model used to describe the system and its specifications.

## 3.1 Computational Model

Application programmers use TransMAN with a set of function calls to send and receive messages from a group, and to detect changes in group view. The simple API hides the complexity of changing group membership and the unreliable communication medium, as elaborated by Birman (1993), Birman & Joseph (1987b) and Birman & Joseph (1987a). The TransMAN API is shown in Table 3.1.

The dynamic nature of MANETs requires that nodes should be able to "discover" groups in a geographical area, allowing applications to later decide to join the group or not. The **discover**() call enables a node to discover any group present in its vicinity. Implementing **discover** requires TransMAN to keep a list of group ids from which it has received broadcast messages. When a node wants to join a group it simply sends a broadcast message using the **send**($m$, $G$) call with the group identifier ($G$) as the destination address.

The use of a broadcast message to initiate a view change or a "join" treats all nodes as equals, the group formations and mergers are therefore treated the same. This results

| Function Call | Description |
|---|---|
| **discover**() | Discovers the groups present in the vicinity of a node. The function returns the list of group identifiers that are operational in the area. |
| **send**($m$, $G$) | Send the message $m$ to the group identified by group identifier $G$. If the sender is not a member of $G$ this message will trigger a group view change at the existing members of $G$. |
| **receive**($m$, $G$) | The node receives a message from a member in $G$ and is guaranteed to be delivered by all nodes that remain in the group $G$. |
| **stable**($m$, $G$) | The message $m$ has been delivered at all nodes in the group $G$. |
| **view_change**($V$, $G$) | The group $G$'s membership has changed to the list of nodes, $V$. |

Table 3.1: TransMAN Group Communication API

in symmetric solutions for merging and forming groups. Such an approach is useful in MANETs that face challenges of frequent connections and disconnections and treating all nodes as equals is important.

Once a new node sends a message to a group, the membership service at the participating nodes runs a group view change protocol to determine the new group view. The set of participating nodes now includes all nodes, both old and new. The distinction between old and new nodes is only made for the purposes of elaborating the protocol. If two nodes are merging into a group both the nodes see the other node as the "new" node.

During the group view change protocol the message that triggered a group view change and the subsequent messages from all the participating nodes are reliably delivered at all the participating nodes. However, while the group view change protocol is in progress messages from non-members are not delivered to the application. The detailed behaviour specifying which messages are delivered at participating nodes and which nodes will be the members of the new group is discussed in Section 3.4. Once the group view change

protocol is complete, the **view_change**($V$, $G$) call is used to inform higher layers of the new group membership.

During normal operations of a group, that is when the group view change protocol is not running, messages are reliably delivered at all group members using the **receive**($m$, $G$) call. Finally the application is informed that a message has been delivered at all nodes in the group view by the call **stable**($m$, $G$). Such a function is required because the **receive** function delivers messages in a reliable FIFO order, but some applications want to be informed when the TransMAN daemon has ascertained that all members have **receive**d the message.

### 3.1.1 Supporting Mobile Groups

The API shown in Table 3.1 is independent of a node's geographical location and group communication is oblivious to location changes of the participants. Independence from node locations makes the system available even in the absence of location services and avoids the extraneous traffic generated by a location dissemination service (Abbadi et al. 1985).

Nodes remain in a group communication as long as all nodes receive messages from each other. Location information is not needed to maintain the group view and therefore TransMAN does not require a location service. However, if a location service is available it can be used to optimise the broadcast layer (Tseng et al. 2002, Sun et al. 2001).

### 3.1.2 Group's Geographical Size

The API presented is independent of the group's size allowing the group to scale to any geographical area as long as they are in the communication ranges of the other nodes in the group. Meier et al. (2001) suggest the use of a bounded geographical area to limit the size of the group. This geographical bounding is used to provide timely communication among the group members but requires a location service to provide the location of nodes to the higher layers. Instead TransMAN uses hop counts to limit the extent of a group, bypassing the need for a location service and forgoing timeliness guarantees in exchange.

## 3.2 Assumptions

This section states the assumptions under which the system is designed and implemented. The assumptions are chosen to allow an implementation of TransMAN to use existing libraries from time sharing operating systems. This requirement rules out assumptions that require special hardware of services like clock synchronisation and location awareness.

**Dynamic Set of Nodes** – It is assumed that the system consists of a dynamic set of nodes where nodes can dynamically join and leave the system and that these nodes communicate using message passing.

**Unreliable Datagram Service** – It is assumed that the underlying network provides an unreliable datagram communication facility between a node and its neighbours. This assumption maps to the wireless communication protocols like IEEE 802.11b.

**Mobility** – Nodes are assumed to be mobile, that is they continue operation while changing their geographic location. Node mobility results in a node's neighbours changing dynamically, which can further cause network partitions and mergers.

**Crash Failures and Recovery** – It is assumed that nodes can crash and recover, but Byzantine failures do not occur.

**Stable Storage** – Nodes are assumed to have stable storage on them.

**Eventual fair broadcast** – In an infinite execution, each node broadcasts messages infinitely often; the immediate neighbours of the node receive infinitely many of those messages; and if a message is rebroadcast infinitely often, all neighbours of the sending node eventually receive that message. This assumption ignores the fairness problem (Barrett, Marathe, Engelhart & Sivasubramaniam 2002) encountered with the IEEE 802.11 MAC protocol.

**No spurious messages** – All messages received by any node have been sent by some participating node. This assumption eliminates the cases where a node is sending messages in the MANET but is not participating in the group communication.

### 3.2.1 Failure Models

Failures in the system model are classified as either node failures or communication failures. A node failure is assumed to fail-stop and no Byzantine failures are handled by the system. A node can crash fail by halting either permanently or temporarily due to running out of battery, switching to a power saving mode or due to other hardware and software failures at the node.

A node can recover from a crash, for example when it comes out of a power saving mode, and is required to identify itself as a new incarnation of the same node. The stable storage on a node allows a recovering node to keep track of its incarnation number.

A crash failure will break the communication between other nodes in MANETs because of the dependence on a multi-hop datagram service. In such a situation, a crash failure may result in a communication failure between nodes that did not crash. A communication failure occurs whenever two nodes can no longer exchange messages either directly or by passing messages through intermediate nodes. A communication failure leads to a set of nodes getting partitioned, where nodes in each partition communicate with each other but not with those in the other partition. Node mobility can also result in nodes losing communication with each other, causing similar partitions.

#### Node Reachability

A node $p$ is defined **reachable** from $q$ if both the nodes can send and receive messages to and from each other either directly or by forwarding messages via intermediate neighbours. Any two nodes are reachable from each other only if bi-directional communication between them is possible. Nodes become **unreachable** from each other if either there is a communication failure between the two nodes caused by (1) crash failures of nodes, or (2) relative movement of nodes resulting in severing of the communication links between them.

Experiences with building Group Communication systems like Isis, Horus (van Renesse et al. 1996), Trans (Moser, Melliar-Smith & Agarwal 1994) and Totem (Amir et al. 1995) noted the problem of a system's inability to distinguish a crash failure from a communica-

tion failure. The problem arises because of the use of timeouts or lack of messages received to determine whether a node has failed. The failure of a communication link between two nodes is interpreted by each node as the failure of the other node. In the case of multi-hop networks, like the 802.11b network, where communication failures are caused by both crash failures and node mobility, the failure scenario is complicated. This thesis, therefore, does not distinguish between a crash failure or a communication failure.

## 3.3 System Model

TransMAN's specifications are based on the framework presented by Chockler et al. (2001), who used the un-timed input/output automaton (IOA) model (Lynch & Tuttle 1989) to specify a set of group communications properties. The input/output automaton model is a tool for modelling components in asynchronous concurrent systems and has been used to model, describe and reason about distributed systems. The framework for the GCS properties described by Chockler et al. (2001) is one of the mature means of specifying a GCS and has developed as a result of a long debate in the community (Babaoğlu et al. 2001, Hiltunen & Schlichting 1995, Babaoğlu et al. 2001, Fekete et al. 2001, Babaoğlu, Davoli & Montresor 1998). Using IOA, the properties of a GCS are presented as a collection of trace properties. A **trace** is defined as a sequence of external actions taken by the IOA.

The specification presented also borrows properties from the framework presented by Babaoğlu et al. (2001), who provide specifications for a partitionable group communication service for WANs. This model is used by Babaoğlu et al. (2001) to specify a group communication system for WANs and is relevant in MANETs because of the modelling of MANETs as overlapping broadcast domains connected via dynamic gateways, which shows a behaviour closer to WANs than LANs. For example, in MANETs as in WANs, a message has to be forwarded by several nodes to reach its destination, which is not the case in a LAN.

This section presents TransMAN's specifications by defining an external signature and the trace properties of the TransMAN IOA. First some useful definitions are introduced:

$\mathcal{P}$ – The set of nodes that can participate in a group. This set is never fully determined and is not used in implementation, but it is useful for formally stating the system properties.

$\mathcal{M}$ – The set of message identifiers.

$\mathcal{VID}$ – The set of view identifiers.

$\mathcal{V}$ – The set of views delivered to the application. A view is a pair containing a view identifier and the set of members in the view.

$T$ – The transitional set used to implement the virtual synchrony properties of the system, explained in Equation (A.12)(A.13) of Appendix A.

Listed next are the set of external actions for the TransMAN IOA. These external actions are called **events**.

**send**$(p, m)$, $p \in \mathcal{P}$, $m \in \mathcal{M}$ – Node $p$ sends a message $m$.

**recv**$(p, m)$, $p \in \mathcal{P}$, $m \in \mathcal{M}$ – Node $p$ receives a message $m$.

**crash**$(p)$, $p \in \mathcal{P}$ – Node $p$ is reported to have crashed. **crash** encapsulates all types of *crash* and *communication* failures.

**recover**$(p, m)$, $p \in \mathcal{P}$ – Node $p$ is reported to have recovered from a crash. The same event occurs if a node that has not been participating in a group enters the radio range of the group.

**view_chng**$(p, \langle id, members \rangle, T)$, $p \in \mathcal{P}, id \in \mathcal{VID}, members \in 2^{\mathcal{P}}, T \in 2^{\mathcal{P}}$ – Node $p$ installs a new view with identity $id$ and members as in the set of nodes $members$ and $T$ is the transitional set from the Extended Virtual Synchrony (Section A.0.2) model.

### 3.3.1 Stability Conditions and Failure Detectors

The group communication services targeted by Chockler et al. (2001) and Babaoğlu et al. (2001) are best effort services where the specified properties are available only when certain stability conditions are satisfied. This section presents the stability conditions required by TransMAN to provide the specifications listed in Appendix A. This section also presents the stable components and failure detectors used by TransMAN, and how these are related to those presented in the specifications.

Babaoğlu et al. (2001) define stability conditions for a partitionable group communication service using abstract properties of a failure detector. These properties are an adaptation of the failure detector properties developed by Chandra & Toueg (1996). The framework developed in (Chockler et al. 2001) provides for stability conditions using a "stable component" which is also defined in terms of abstract failure detector properties. Both the stability conditions require that (1) no processes crash or recover, (2) communication becomes symmetric and transitive, and (3) no changes occur in network connectivity.

Observing the presence of these conditions in MANETs is not straightforward. Consider the case where nodes $p$ and $q$ are in the same partition and some nodes move from the neighbourhood of $p$ to the neighbourhood of $q$. Three difficulties emerge: (1) the nodes $p$ and $q$ have differing observations about nodes that move from $p$'s neighbourhood to $q$'s, resulting in a conflict with the "no nodes crash and recover" condition, (2) transitive and symmetric communication can not be guaranteed, and (3) changes in the immediate reachability contradict the "no changes in network connectivity" condition. The work presented here does not assume that the above stability conditions will be detected by nodes but instead, following the lead from Chockler et al. (2001) and Babaoğlu et al. (2001) specifications frameworks, the stability conditions are specified by requiring certain properties from the system's failure detector.

#### Failure Detector

A failure detector is a distributed oracle with an instance running at each participating node that maintains a list of nodes suspected by each of these nodes. The failure detector

can make mistakes and once it realises a mistake it can correct itself. As discussed in Section 3.2.1 node mobility and crash failures result in communication failure because of the mobile and multi-hop nature of MANETs. This section shows how these failures affect failure detection in MANETs. In particular, the discussion shows how the completeness and accuracy properties of the failure detectors are affected by node mobility.

The completeness property of a failure detector pertains to detecting a failure and informing the nodes in the system about this failure. The accuracy property of a failure detector pertains to detecting a correct process, i.e., a process that has not failed. Chandra & Toueg (1996) define these properties as –

**Completeness** – There is a time after which every process that crashes is permanently suspected by some correct process.

**Accuracy** – There is a time after which some correct process is never suspected by any correct process.

Traditional group communication services provide a failure detector by using a pinging service. For example, in RELACS (Babaoğlu, Davoli, Montresor & Segala 1998) each node pings every other node it is aware of and determines failures based on time outs for responses to the pings. A direct port of such a service to mobile ad-hoc networks requires IP routes from all nodes to all other nodes. Maintaining IP routes between nodes in a mobile ad-hoc network requires a routing service that generates its own traffic to maintain routing information, wasting the limited battery life of nodes.

An alternative approach for providing failure detectors in MANETs is to provide a local failure detector module at each node that keeps track of its neighbours. This can be done by using any of the neighbour discovery protocols available in protocols like AODV (Perkins et al. 2003) and OLSR (Jacquet et al. 2003). Using a neighbour discovery service improves the scalability of the failure detector and will work without the need for maintaining knowledge about all the nodes in the system.

However, there are two main problems with the use of such a neighbourhood-based failure detection service. The first problem occurs because of node mobility. Consider

the example when a node $r$ moves from $p$'s neighbourhood to $q$'s neighbourhood, $p$ will detect this as a failure and will inform the other nodes of $r$'s failure. At the same time, $q$ will detect a "new" node and will disseminate this information. Resolving such conflicting information about the state of processors requires sophisticated schemes that will generate extraneous traffic in the MANET. The second problem occurs in the case of partitions. Consider two disjoint subsets of nodes, $L$ and $M$, connected via one hop through node $p$ and where nodes $r \in L$ and $q \in M$ are in the neighbourhood of node $p$. If $p$ fails, $r$ and $q$ detect $p$'s failure and will inform the nodes in their corresponding partitions. The problem is that none of the local failure detector components running in $L$ and $M$ are able to detect the failure of the rest of the nodes in the other partition.

The failure detector used in TransMAN does not rely on maintenance of routing information in a MANET or tracking neighbours of a node. Instead, TransMAN utilises application broadcast messages to suspect node failures. While a node $p$ receives messages from another node $q$, $p$ does not suspect $q$. The use of application messages to implement a failure detector eliminates the need for generating extraneous control messages or implementing sophisticated schemes to track neighbours. The failure detector is then used to trigger the membership agreement protocol that installs a new agreed-upon group view at nodes that survive the group view change. Chandra, Hadzilacos & Toueg (1996) show the weakest failure detector required to solve agreement in asynchronous networks can make a lot of mistakes. Section 4.4 describes the design of the failure detector provided by TransMAN in detail and shows how it is at least as strong as the weakest failure detector required to solve agreement.

Even though the failure detector used in TransMAN is strong enough to help solve agreement, it is important to note that TransMAN does not face the impossibility results (Fischer, Lynch & Paterson 1985) that generate the need for using a failure detector to solve agreement. This is because TransMAN allows groups to partition and starts a new agreement protocol if there are any changes to the list of participants in the network. The impossibility result (Fischer et al. 1985) is applicable only to networks that do not allow partitions.

## 3.4 Specifications

Appendix A lists the specifications for TransMAN that have been borrowed from the frameworks presented by Chockler et al. (2001) and Babaoğlu et al. (2001). The specifications meet the system requirements presented in Section 1.1 on page 1.

The mappings between the system specifications and system requirements are shown in Table 3.2. These mappings show that a number of properties are required to meet each of the guarantees. The properties that form the specifications provide a smaller unit of reasoning that is more tractable than the system requirements. Later chapters prove that TransMAN provides these individual specifications.

| **System Requirement** | **Property(Equation)** |
|---|---|
| Reliable Broadcast<br>Ordered Delivery | Delivery Integrity (A.8)<br>No Duplication (A.9)<br>Self Delivery (A.10)<br>Message Ordering (A.14)(A.15)(A.16) |
| Consistent List of Participating Nodes<br>Non-Blocking View Changes | View Coherency (A.6)(A.7)<br>Local Monotonicity (A.2)<br>Self Inclusion (A.1)<br>Initial View Event (A.3)<br>Membership Accuracy (A.4)<br>Termination of Delivery (A.5) |
| Virtual Synchrony | Transitional Sets (A.11)(A.12)(A.13) |

Table 3.2: Mapping System Requirements to Specification Properties

### 3.4.1 Conclusions

The properties listed in this chapter specify TransMAN as a partitionable group communication system for MANETs with virtual synchronous communication. The specifications allow nodes to continue to broadcast messages while a group view change is underway.

A non-blocking approach for virtually synchronous group communication is important in MANETs because of the transient network connections expected in MANETs. A mapping of the system specifications to system requirements was presented, allowing for smaller units of reasoning that are then used to provide the system requirements.

# Chapter 4

# A Virtually Synchronous Group Communication for MANETs

This chapter presents the design of a reliable broadcast protocol and a partitionable membership service that constitute TransMAN. Algorithms for implementing a shared message space across overlapping broadcast domains are presented. The shared message space is a directed acyclic graph, and is used to implement message stability and membership agreement. The chapter also presents proofs for the correctness of the algorithms used to implement each of these services. The pseudocode for the algorithms are shown in Appendix C.

## 4.1 System Architecture

Figure 4.1 illustrates the system architecture for TransMAN. This lowest layer is the broadcast protocol, which uses existing broadcast optimisations for MANETs. The reliable broadcast layer delivers messages in a FIFO order (Section 4.2). These FIFO deliveries may be used directly by the application or may be queued for stabilisation. The message stability layer (Section 4.3) delivers messages in a total order and these messages are guaranteed to be delivered by all the nodes participating in the group. Messages delivered in FIFO order are used to suspect node failures (Section 4.4) and reach agreement on group

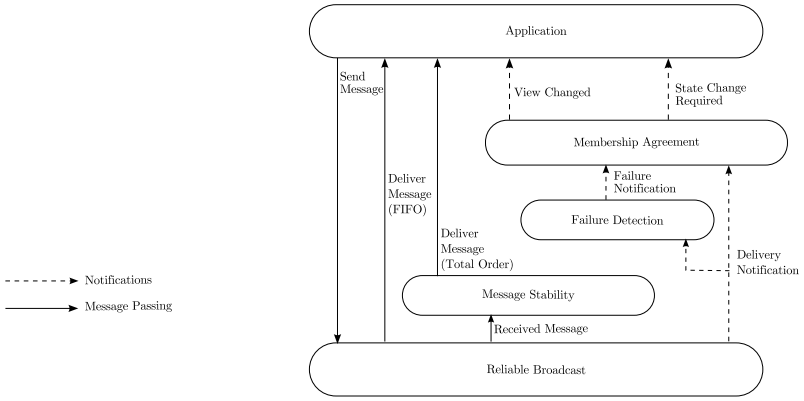


Figure 4.1: System Architecture

membership (Section 4.5.1). The membership agreement layer updates the application proactively if there are any changes in the group. The application sends messages through the broadcast layer, which communicates the required updates to the failure detector and membership agreement layers.

## 4.2 Reliable Broadcast

Reliability is implemented using negative acknowledgements (nacks). A node that realises it has missed a message sends a negative acknowledgement for the message. A transmitting node assumes that the messages it has broadcast are received at its neighbours. This results in an optimistic approach for sending messages whereby a node keeps transmitting messages assuming the earlier messages have been received by one or more of its neighbours. This approach supports the implementation of a group membership service that does not block applications from sending messages during view changes. The approach also addresses the lack of collision detection in wireless medium access control protocols.

Before describing the reliable broadcast protocol in detail some terminology is introduced. A message is *sent* by a node if the node originates the message. The sender of a message is therefore the first node to transmit the message. A message is said to be *forwarded* by a node if it receives a message sent by some other node and then retransmits it. A message is said to be *delivered* at a node if it has been received by the node

and after fulfilling certain conditions is ready to be handed to the application at the node. These conditions are described in Section 4.2.2 on page 64.

To provide reliable broadcast, the protocol attaches a header to each message that is broadcast. The header includes the message identifier (*mid*) of the message and the dependencies of the message. The dependencies capture the relationships between messages and are introduced in the next section. They are used to implement the shared messages space that is used to provide stability and message agreement. A *mid* is composed of the *id* of the sending node and the *sequence number* of the message; thus $p_i$ is a message sent by $p$ with sequence number $i$.

The procedure CreateNewPacket in Module 2 on page 147 (Appendix C) shows these dependencies being included in the header of messages broadcast. Messages are created to include the dependencies and the broadcast layer sends timeout messages when there are no application messages (as shown in procedure SendThread of Module 2) The timeout messages are central to the TransMAN GCS because TransMAN assumes a continuous series of messages from all participating nodes. This is discussed further during the presentation of the failure detector and membership agreement protocols.

### 4.2.1 Message Dependencies

Message Dependencies capture relationships between messages that are used to provide reliable broadcast. The key idea is that a message is not delivered by a node until all its dependencies are delivered. If a message can not be delivered negative acknowledgements are sent for the missing dependencies. This approach enables FIFO delivery of messages and this section shows the design for this reliable FIFO service.

Message dependencies are also used to implement the shared message space concept presented by Peterson et al. (1989) and the implicit acknowledgement concept by Melliar-Smith & Moser (1993). The use of a shared message space allows for a group membership service that does not need to block applications during view changes and can simultaneously run agreement on a number of next possible views. These features are essential in a MANET where the dynamic nature of the network generates a number of frequent transient

connections and disconnections.

**Message Dependency** – A message $p_i$ is said to be dependent on a message $q_j$ if $p$ receives and delivers $q_j$ before sending $p_i$ or if $q = p \wedge j = i - 1$. This dependency relationship is transitive.

There are two kinds of dependencies: "last sent" and "last delivered". Both these dependencies together help implement reliability and the shared messages space.

**Last Sent Dependency** – If p sends a message $p_i$, $p_{i-1}$ is a dependency of $p_i$, and is called the last sent dependency of $p_i$.

**Last Delivered Dependency** – If a node $p$ receives and delivers a message $q_k$ before sending a message $p_i$, $q_k$ is the last delivered dependency of $p_i$, if $p$ does not deliver any messages between delivering $q_k$ and sending $p_i$.

Message dependencies are transitive and are used to provide implicit acknowledgements for message delivery. If a message $m$ is a dependency of $n$, the dependency relationship is written as $m \rightarrow n$. A single letter like $m$ is used to identify a message if the sender of the message is not important to the discussion.

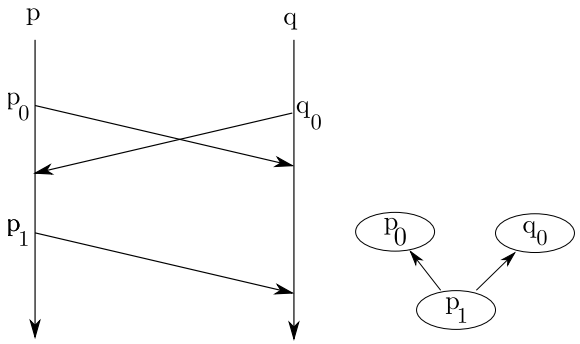


Figure 4.2: Message Dependencies

Figure 4.2 shows two nodes $p$ and $q$ exchanging messages $p_0, p_1$ and $q_0$. The timeline diagram shows $p$ and $q$ concurrently broadcast messages $p_0$ and $q_0$, which are the first messages broadcast by their senders and therefore the dependencies in the headers are set

to *null*. However, when $p_1$ is broadcast, its last sent dependency is $p_0$ and last delivered dependency is $q_0$. Both these dependencies are shown on Figure 4.2 to the right. The arrows show the dependency relationship between messages that are shown in eclipses. This notation of using arrows to show dependency relationship is used throughout this thesis.

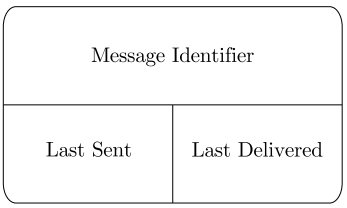


Figure 4.3: Message Header

Figure 4.3 shows the reliable broadcast message header. The "Message Identifier" is used to uniquely identify the message and is composed of the sender address and the sequence number of the message. As described the "Last Sent" and "Last Delivered" fields are the dependencies of the message and are used to provide reliability and to implement the shared message space that is used to determine message stability (Section 4.3) and membership agreement (Section 4.5.1).

The message header has a constant size and is unaffected by the presence of other nodes in the neighbourhood of the sender. These features are useful when the reliable broadcast is used to implement a partitionable group membership service. Message size remains independent of the group size avoiding the growth in message size as the group size increases. The independence of the reliable broadcast from knowledge of the network also allows the reliable broadcast at any node to behave the same, independent of network conditions and of services that gather information about the network like location service.

Reliable broadcast is provided by using the transitive relationship between the dependencies defined above. To ease the description of the protocols that follow the transitive relationship relationship between dependencies is used to define the notion of "transitive dependencies".

**Transitive Dependency** – If $m \rightarrow n \wedge n \rightarrow \ldots \rightarrow o$ then $m$ is called the transitive

dependency of $o$ and is written as $m \triangleright o$.

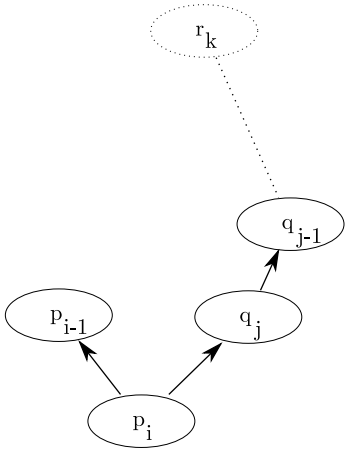


Figure 4.4: Transitive Dependencies

Figure 4.4 shows dependencies between messages $p_i, p_{i-1}, q_j, q_{j-1}$ and $r_k$. The last sent and last delivered dependencies of $p_i$ are $p_{i-1}$ and $q_j$. Both the dependencies of $p_i$ are shown but only the last sent dependency of $q_j$ is shown, which is $q_{j-1}$. Therefore, $q_{j-1}$ is a transitive dependency of $p_i$, i.e., $q_{j-1} \triangleright p_i$. Another message $r_k$ is shown as $r_k \triangleright q_{j-i}$, which implies $r_k \triangleright p_i$. The definitions lead to the following lemma:

**Lemma 1.** *A message $m$ delivered by a node $p$ becomes a transitive dependency of any message $p_k$ that is sent by $p$ after delivering $m$.*

*Proof.* From the definition of dependencies, the first message $p_i$ sent by $p$ after delivering $m$ includes $m$ as a dependency; therefore by the definition of transitive dependency, all messages $p_j$ sent by $p$ such that $j > i$ have $m$ as an transitive dependency. □

The dependencies and transitive dependencies are central to the protocols presented in this thesis. The dependency relationships between messages are used to provide reliable broadcast, message stability determination and group membership agreement.

### 4.2.2 Reliability Via Dependencies

Reliable broadcast requires that a node is able to identify situations where it has lost messages and is then able to recover the lost messages. Message dependencies are used by

a node to identify if any messages have not been received by it. The key idea is that a messages is delivered only if all its transitive dependencies have been delivered. The two fundamental rules for providing reliable broadcast are –

**$\mathcal{R}$1** A message received by a node is not delivered until both its last sent and last delivered dependencies are received and delivered at the receiving node.

**$\mathcal{R}$2** A message is not forwarded by a node until it has been delivered at the node.

The following lemma follows from the above rules and is useful to prove the correctness of the reliability protocol:

**Lemma 2.** *A node $p$ delivers a received message $n$ only if all messages $m$ such that $m \triangleright n$ have been delivered by $p$.*

*Proof.* Follows from the definition of transitive dependency and the rule $\mathcal{R}1$. □

**Corollary 3.** *If a node $p$ delivers a message $n$ it has delivered all messages $m$ where $m \triangleright n$.*

### Negative Acknowledgements

$\mathcal{R}1$ allows for nodes to determine if they have missed a message. A node recovers the missing message by sending negative acknowledgements (nacks). $\mathcal{R}2$ guarantees that if a node sends a negative acknowledgement to its neighbours, one or more of them have a copy of the message. This section describes how negative acknowledgements are generated when a node receives a message.

Suppose a node $p$ receives a message $q_i$ with two dependencies as $q_{i-1}$ and $s_j$. From rule $\mathcal{R}1$, $p$ will deliver the message only if $p$ has received and delivered both $q_{i-1}$ and $s_j$. If $p$ can not deliver $q_i$, it is because some transitive dependency of $q_i$ has not been received by $p$. This requires $p$ to nack for the missing transitive dependency of $q_i$.

To find the transitive dependency to nack for, $p$ checks if it has received either of the two dependencies of $q_i$, that is $q_i - 1$ and $s_j$. If $p$ has not received $q_{i-1}$ and $s_j$, $p$ nacks for the missing dependency. If it has received the dependencies, $p$ recursively checks for

all the transitive dependencies of $p$ until it finds a message that has not yet been received. This leads to the third rule for the reliable broadcast protocol.

**$\mathcal{R}3$** For every message received that cannot be immediately delivered, the receiving node sends a nack for some transitive dependency of the received message that has not yet been received.

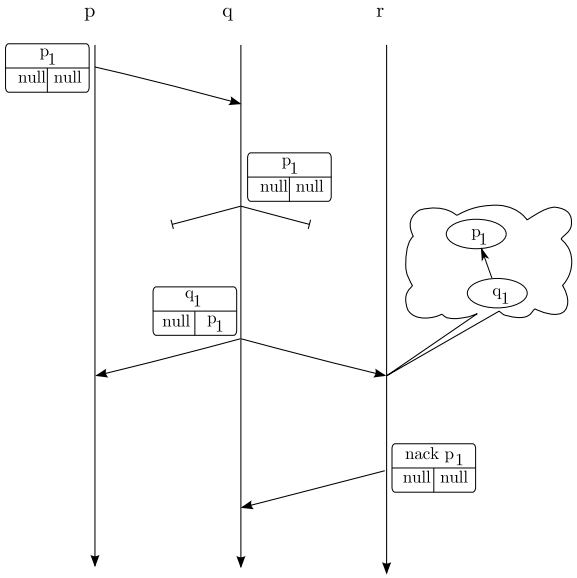


Figure 4.5: Negative Acknowledgements for Message Dependencies

Figure 4.5 shows three nodes $p, q$ and $r$ exchanging an initial set of messages which leads to $r$ sending a negative acknowledgement for the message $p_1$. The message $p_1$ is forwarded by $q$, but is lost during its transmission. However, when $r$ receives the message $q_1$ it is able to determine the loss of message $p_1$ and sends a negative acknowledgement for it.

This approach of providing reliable broadcast requires that nodes send messages at regular intervals to help other nodes determine missing dependencies. A lost message is recovered only if some message is received that has a transitive dependency relationship to one of the lost messages. The series of messages is then forwarded by nodes to enable nodes more than one hop away from the sender of a message to receive and deliver the

message. If the application at a node is not sending any messages, the TransMAN daemon sends timeout messages to facilitate the progress of the protocol on all other nodes and also to avoid the other nodes from removing this node from their group view (see Section 4.4). This leads to the fourth rule for the protocol.

$\mathcal{R}4$ Every message that is received and delivered by a node is also forwarded by the node.

The regular stream of messages sent by each node makes the system best suited for applications that require all-to-all streams of regular messages, like mobile multiplayer games and collaborative work. From the pseudocode presented in Appendix C, the procedure SendThread in Module 2 on page 147 shows how this stream of messages is generated when no application messages are present.

Modules 3 to 5 on pages 147–149 of Appendix C show how nodes determine lost messages when they receive new messages, and how nodes send negative acknowledgements for the missing messages. The procedure SendNacksFor (Module 5) shows the protocol for determining the missing dependencies of a message. Once the missing dependencies are recovered, the messages are delivered as shown in the procedure Deliver (Module 4). Nodes that receive a negative acknowledgement respond with the missing message, if they have it, as shown in the procedure RespondToNack.

### Reliability Proofs

Rules $\mathcal{R}2$ and $\mathcal{R}4$ work together to make sure a message is forwarded if it is delivered and is not forwarded until it is delivered. The property that guarantees dissemination of messages because of these rules is presented next.

**Eventual Reception Property** – If any node $p$ broadcasts or receives a message, then every node that remains reachable from $p$, eventually receives the message.

To establish this property Theorems 4, 5 and 6 are presented.

**Theorem 4.** *Given two nodes $p$ and $q$ remain neighbours, if $q$ receives a message $m$ forwarded by $p$, $q$ eventually delivers the message.*

*Proof.* From Corollary 3, $p$ has received and delivered all transitive dependencies of $m$, and from rule $\mathcal{R}3$, $q$ nacks for missing dependencies of $m$. Given $p$ and $q$ remain neighbours and the eventual broadcast assumption, $q$ eventually delivers $m$. □

**Theorem 5.** *Every message sent by a node $p$ is eventually received by every node that remains reachable from $p$.*

*Proof.* The proof for this theorem is presented in Appendix B. □

**Theorem 6.** *Every message received by a node $p$ is eventually received by every node that remains reachable from $p$.*

*Proof.* When $p$ receives the message $m$, by Theorem 4, $p$ delivers $m$ and by rule $\mathcal{R}4$, $p$ forwards $m$. From the no spurious messages assumption, $m$ has been broadcast by some participating node. Finally, from Theorem 5 all nodes that remain reachable from $p$ receive $m$. □

## 4.3 Message Stability

The dependencies and many-to-many reliable broadcast help implement a useful facility called the "Observable Predicate for Delivery", or OPD. The OPD allows a node to determine if another node has yet received and delivered a message sent by any of the participating nodes. The OPD is then used to determine message stability and membership agreement. This section first describes OPD and its use for determining message stability. This section then describes an algorithm that uses OPD for stabilising messages in a total order.

### 4.3.1 Observable Predicate For Delivery

The *Observable Predicate for Delivery*, or *OPD*, allows a node $p$ to determine if the sender of a certain message $q_i$ has received and delivered a message $r_j$ before sending $q_i$. $OPD(p, r_j, q_i)$ is said to be true if node $p$ can determine that the node $q$ has delivered the message $r_j$ before $q$ sent $q_i$. The Trans broadcast system (Moser, Melliar-Smith &

Agarwal 1994) first employed the property to build a partial relation between messages and to determine message stability. The Trans protocols then converted the partial order relationship to a totally ordered one and implemented a membership protocol using the total order of messages. TransMAN does not convert the partial order to a total order straightaway as the Trans approach. This is because the rate of message losses in MANETs is expected to be much higher than those in a LAN and the Trans approach assumes a very low rate of message losses. TransMAN instead utilises the partial order between messages to determine message stability and provide membership agreement protocol. This approach balances the opportunity of using the MANET broadcast domain while avoiding total ordering approaches used in LANs in recognition of the high error rate expected in MANETs.

For the purposes of evaluating the OPD, TransMAN maintains a directed graph of message identifiers at each node. This graph is called the "Delivered Before Graph" (DBG). A message identifier for message $s_k$ sent by $s$ has two edges to it in the DBG. These represent $s_k$'s dependencies, i.e., last sent and last delivered. Trans and Psync construct a similar graph of messages but they do not use these dependencies. Instead, implicit transitive positive acknowledgements are used to implement the shared message space (for a detailed discussion see Section 2.3.1).

The main idea for determining the OPD is based on the transitivity relationship in message dependencies and the way the DBG is constructed. DBGs have a path from a message $m$ to $n$ if and only if $m \triangleright n$. From rule $\mathcal{R}2$, if $m \triangleright n$, we know that the sender of $n$ is guaranteed to have delivered $m$ before sending $n$. These statements are later proved as Theorem 7, but first the conditions under which $OPD(p, r_j, q_i)$ evaluates to true are presented. The $OPD(p, r_j, q_i)$ is true if –

1. $p$ has delivered a sequence of messages starting from message $r_j$ and ending in message $q_i$, and

2. there is a path from $r_j$ to $q_i$ in $p$'s DBG.

Figure 4.6 shows an example where nodes $p$ and $q$ broadcast messages and $r$ is able

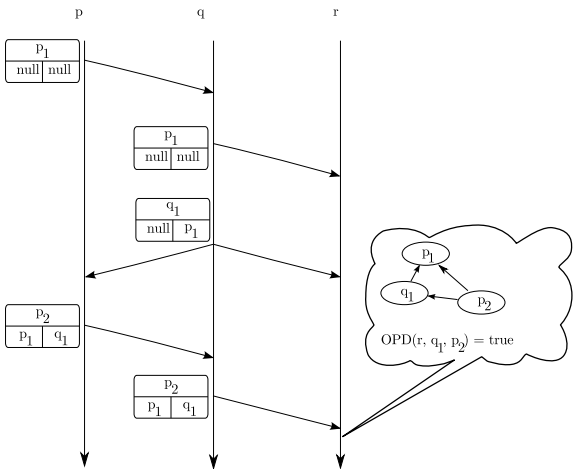


Figure 4.6: Evaluating OPD - An Example

to determine if $p$ has delivered message $q_1$ when $p$ sent $p_2$. To ascertain this, the node $r$ evaluates the OPD$(r, q_1, p_2)$ on delivering the message $p_2$. The OPD evaluates to true because there is a path from $q_1$ to $p_2$ in $r$'s DBG (shown in the cloud).

**Theorem 7.** *If there is a path from message $p_i$ to $q_j$ in a node $r$'s DBG, then $q$ has received and delivered $p_i$ before broadcasting $q_j$.*

*Proof.* From the the definition of DBG, $p_i \triangleright q_j$, and from rule $\mathcal{R}2$ $q$ has delivered $q_j$. Finally, from Corollary 3, $q$ has delivered $p_i$. □

### 4.3.2 Stability Protocol

A message is said to be "stable" when it has been delivered at all the participating nodes. In the case of a group communication service, the participating nodes are the nodes in a group view. TransMAN deems a message as stable when it is established that it has been delivered at all nodes in the current group view. Each node independently determines the stability of a message by using the message dependency relationships and the DBG.

For the purposes of explaining the stability protocol the membership of the mobile ad-hoc network under consideration is assumed to be known as the set of nodes, $\mathcal{M}$. The group membership is also assumed to remain the same during the system run. This assumption is only required in the absence of a membership service. Once a membership service is

available (Section 4.5) this assumption about a-priori knowledge of group membership is not required.

Given the OPD can be evaluated at all the participating nodes, determining message stability is straightforward. On reception of a message, the receiving node uses the OPD mechanism to evaluate if any of its set of unstable messages is now stable. A data structure maintained by the participating nodes is used:

"Message Last Delivered" ($mld$) is a list maintained at each node that contains references to the messages last delivered from all other nodes in $\mathcal{M}$. $mld[p]$ at node $q$ is the message last delivered from $p$ at node $q$. Given the definitions of $mld$ and OPD the following lemma arises:

**Lemma 8.** *A message $q_i$ is determined as stable at node $p$ when $\forall n \in \mathcal{M}$ $OPD(p, q_i, mld[n]) = true$.*

*Proof.* The proof follows from the definition of OPD. When the condition in the lemma is true, $p$ knows that the message $q_i$ is delivered at all nodes participating in the broadcast, and thus $q_i$ is stable at $p$. □

The proof for the liveness of the stability protocol is presented next. Liveness guarantees that eventually the system provides the guarantees it specifies. In the case of the stability protocol, the liveness property is stated as:

**Theorem 9.** *In an infinite execution of the protocol, when nodes remain reachable from each other, a message $q_i$ eventually becomes stable at all participating nodes.*

*Proof.* From Theorem 6, in an infinite execution the message $q_i$ is eventually received at all participating nodes. Further from the eventual broadcast assumption, all participating nodes will broadcast messages which by Lemma 1 have $q_i$ as a transitive dependency. Again from Theorem 6, these messages will be received at all other nodes, thus resulting in $q_i$ being determined stable at all participating nodes. □

The message stability approach involves checking for paths in the DBG after delivering every message. Module 6 on page 150 (Appendix C) shows the pseudocode for determining

message stability. The algorithm is executed at delivery of every message and therefore is a computationally intensive exercise. However, this is a good trade off for the reduction in the number of message transmissions when compared with approaches that require explicit acknowledgements.

### 4.3.3 Total Order

Once a message has stabilised, it can be safely delivered to applications. Applications can use the contents of the messages assured that all other nodes in the group have also delivered the message. In TransMAN, messages are stabilised in a total order across all members of a group because of the way DBGs are constructed at the nodes.

Edges in a DBG are the dependencies between messages. If the DBG is traversed with an a-priori known and unambiguous order, it will generate the same sequence of messages at each node. This is because the DBG captures the same dependency relationship between messages.

However, due to asynchronous communication and a weakly connected network, nodes deliver messages at different rates. This means the DBG at each node is constructed in different orders. In the presence of such variations in the construction of the DBG there is a need to traverse the DBG in an a-priori known order. The following rule provides such an order.

$\mathcal{R}5$ A message is stabilised only if all messages preceding the message in the topological sort of the DBG are already stabilised. Ambiguities in the topological sort are handled by ordering along the message identifiers.

Figure 4.7 on the next page shows an example where DBGs at the participating nodes grow at different rates. The example shows DBGs at nodes $p, q$ and $r$ constructed by exchanging some sequence of messages. The Figure shows that $p$ has delivered messages $[p_1, p_2, p_3, p_4, q_1, q_2, q_3, r_1]$ while $r$ has delivered $[p_1, q_1, q_2, r_1, r_2, r_3]$ resulting in DBGs constructed up to different depths.

Applying rule $\mathcal{R}5$ to the scenario in Figure 4.7 on the following page, the messages will be stabilised in the same order at all nodes. This is shown now. In the Figure only $r$ is

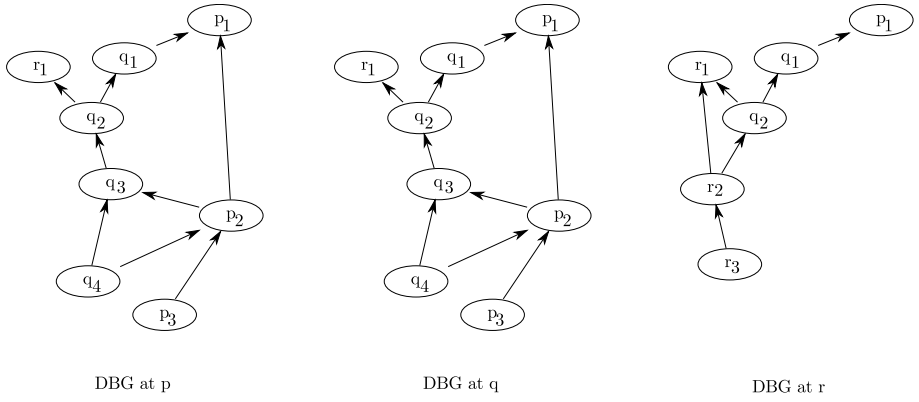


Figure 4.7: DBG of three nodes $p, q$ and $r$. $p_1$ is the first message stabilised at node $r$.

able to determine the stability of a message, i.e. message $p_1$, which is also the first message to be stabilised in the system. This can be seen from the DBG at node $r$, using which $r$ is able to determine that $p$ has delivered $p_1$ (because it is sent by $p$) and $q$ has delivered $p_1$ because $q_1$ has a dependency on $p_1$. Therefore, $r$ concludes that $p_1$ is delivered at all nodes in the network and therefore is stable. The Rule $\mathcal{R}5$ also requires that $p_1$ will be the first message stabilised at all nodes because $p_1$ is the first message in the topological sort of all DBGs. Unless $p_1$ is determined as stable, no other message will be checked for stability. Finally, the Figure shows that $p_1$ will be stabilised only if $r_2$ is delivered at all nodes, this is because $r_2$ acknowledges the delivery of $p_1$ at $r$, and $r_2$ comes later in the topological sort of all the DBGs as compared to $p_1$ and $q_1$. Theorem 12 proves these claims, with two supporting lemmas:

**Lemma 10.** *It is not possible to stabilise a message $r_m$ at node $p$ if there is some message $q_n$ that precedes $r_m$ in the topological sort of DBG and is not delivered by $p$.*

*Proof.* If a message $q_n$ is not delivered at $p$ then no message $q_l$, such that $l > n$ can be delivered from $q$ such that $\text{OPD}(p, q_l, r_m)$ is true. This follows from rule $\mathcal{R}1$. □

**Lemma 11.** *If all messages up to a depth $d$ of the DBG have been delivered, a topological sort with elimination of ambiguities will give the same sequence of messages at all nodes.*

*Proof.* Follows from the definition and construction of the DBG. □

**Theorem 12.** *Stabilising messages according to $\mathcal{R}5$ provides a total order of stabilised messages across all members of a group.*

*Proof.* Assume the group membership is known and remains the same during the system run. Lemma 10 states that all messages preceding a message $m$ must be delivered before $m$ can be stabilised. This implies that no vertex $v$ will be added to DBG such that $v$ precedes $m$ in the topological order. From the above implication, Lemma 11 and rule $\mathcal{R}5$, all nodes will stabilise messages in the same sequence. □

## 4.4 Failure Detection

Maintaining group membership information in an asynchronous network, even with a single partition, faces the well known result (Chandra et al. 1996) on the impossibility of a group membership protocol for non-partitionable groups in an asynchronous network. Because TransMAN is a partitionable group membership service, the impossibility does not apply to TransMAN. But a failure detector is still required so that the membership service can start suspecting a node as failed, and initiate the view change protocol. For this purpose, a failure detector is included in TransMAN.

The failure detector uses a stream of messages broadcast by each node to start suspecting nodes. Given that each node maintains a membership view, the nodes have the necessary information to detect a member node as failed. There are two ways a node can be suspected by other participating nodes. In the first case, if a node does not receive a message from a certain node while receiving messages from all other nodes in the group, it suspects the other node as failed. A node $p$ detects a node $q$ as failed if $p$ does not receive a message from $q$ while it receives $w$ messages from all other nodes taken together. Such failures are termed "failure to broadcast".

The second approach to detect failures allows nodes to determine if another node is receiving its messages or not. A node $p$ suspects another node $q$ as failed if $p$ broadcasts $w$ messages, from $p_n$ to $p_{n+w}$, and receives no message $q_i$ such that $\mathrm{OPD}(p, p_n, q_i)$ is true.

In other words, $p$ has not received any message from $q$ from which $p$ can determine the reception of any of the last $w$ message broadcast by $p$. Such failures are termed "failure to receive".

The parameter $w$, used in the above approaches, is called "wait length". The wait length can be made either the same on all group members or can be different at each node depending on the number of neighbours of the nodes. In the evaluation study (Chapter 6), the wait length is proportional to the size of the group membership.

Figures 4.8 and 4.9 show the *DeliveredMessages* array that contains the *mid*s and the indexes into the array that are important to this discussion. The failure detector at node $p$ detects node $q$ as "failed to broadcast" if $p$ has delivered $w$ messages none of which were sent by $q$, as shown in Figure 4.8 and in pseudocode (line 10 of Module 7 on page 151, Appendix C). Formally, $p$ suspects $q$ as failed to broadcast, if at $p$:

$$\begin{aligned} \exists\, a, b \in I,\ a = b + w\ \wedge delivered[a] = m \\ \wedge\ delivered[b] = q_i\ \Rightarrow \\ \nexists x \in I,\ b \leq x \leq a\ s.t.\ sender(delivered[x]) = q \end{aligned} \tag{4.1}$$

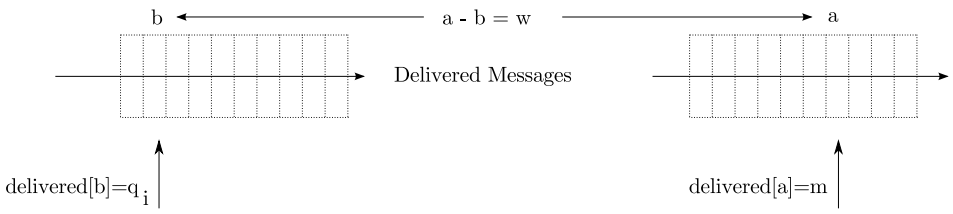


Figure 4.8: Failure To Broadcast

"Failure to receive" is a little more complicated to determine. The failure detector at node $p$ detects node $q$ as "failed to receive" if $p$ has sent messages up to sequence number $n$ and $p$ receives a message $q_i$ such that $q_i$ does not acknowledge the reception of $p_{n-w}$. $p$ can determine this by checking the predicate, $OPD(p, p_{n-w}, q_i)$, as shown in Figure 4.9 and in pseudocode (line 13 of Module 7 on page 151, Appendix C). If the $OPD$ is false $p$ detects $q$ as failing to receive messages from $p$. Formally, $p$ suspects $q$ as failed to send, if

at $p$:

$$\begin{aligned} \exists a, b \in I,\ a > b \wedge delivered[a] = q_i \wedge \\ delivered[b] = last\_sent = p_n \Rightarrow \\ \nexists x \in I,\ b \leq x \leq a\ s.t.\ sender(delivered(x)) \neq q \\ \wedge\ opd(p, p_{n-w}, q_i) = true \end{aligned} \tag{4.2}$$

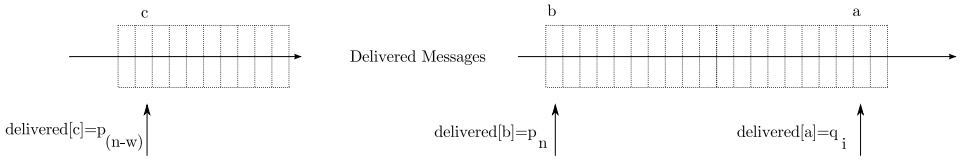


Figure 4.9: Failure To Receive; OPD$(p, p_{n-w}, q_i) = false$

### 4.4.1 Properties

The failure detector supports strong completeness and eventually weak accuracy. If a node fails, the reliable broadcast protocol guarantees that the nodes that have not failed will not deliver any messages from the failed node in future. The protocol also guarantees that the correct nodes will continue to deliver messages from each other. Eventually, the correct nodes will deliver $w$ messages each, and will start suspecting the failed node, guaranteeing a *strong completeness* behaviour of the failure detector.

If a node is suspected when it has not failed, the suspected node will continue to broadcast messages. The broadcast protocol assures that these messages reach all nodes that are still connected to this suspected node. Eventually all nodes will receive at least one message from the suspected node, and will stop suspecting it. This guarantees that eventually no nodes will be falsely suspected by any participating node. This is the *eventual weak accuracy* property of the failure detector.

## 4.5 Membership Agreement

TransMAN uses a membership agreement protocol to provide a virtually synchronous GCS for MANETs. A virtual synchronous communication requires that all nodes that survive the same group view change deliver the same set of messages. The approach to use an agreement protocol is motivated by the work presented by Malloth & Schiper (1995) and Schiper & Ricciardi (1993) where a view synchronous membership service is reduced to an agreement between the participating nodes. The key idea of the approach is that participating nodes reach an agreement on the next group view that includes not only the set of nodes in the new group but also the set of messages these nodes have delivered. This requirement helps provide the guarantee that all nodes that survive the same group view changes deliver the same set of messages. A node refuses agreement on a group view for which it has not delivered all the messages that the view includes as delivered.

The inclusion of delivered messages in a group view helps the agreement protocol by allowing group views to be partially ordered, which allows membership protocol to determine the relative order of proposed views and make informed decisions about ignoring some proposed group views that are no longer viable. Between two nodes that include the same set of nodes, a view that includes messages with higher sequence numbers is considered to be a later view. The *precedence relation* describes the partial ordering relation.

The agreement protocol does not use a coordinator through which all messages are channelled. This avoids the problem where the coordinator requires a higher battery life to transmit all the messages broadcast in the network. Using a totally distributed agreement protocol also avoids the need to repeatedly reelect a coordinator when the network faces transient network partitions and mergers, as expected in a MANET.

The membership agreement protocol is designed to handle transient network connections in a MANET. Every change in network connectivity initiates a membership agreement protocol and if such a change persists then the agreement terminates and the current view at the node is changed to the new group view. To allow for this the protocol requires that each node maintain a list of tentative views. This means that each node considers a number of group views as the next possible view. If the change in network conditions

is transient the agreement protocol does not terminate and is subsequently deleted in the future.

The membership agreement uses application messages (or timeout messages in the absence of application messages) to reach agreement on group views. The use of application messages results in a system where applications are not blocked during a group view change. Instead, the messages transmitted by applications are useful in reaching agreement on the next group view.

The membership service described is a best-effort service. The service continuously strives to install group views reflecting the state of the MANET. If the network faces a high amount of churn, the nodes will work towards installing a group view. As the amount of churn reduces, the nodes install a consistent group view reflecting the current state of the network. The changes to message delivery (if any) during the group view change are discussed in Section 4.5.3.

The following terminology is used to describe the membership agreement protocol used by TransMAN.

**Group view,** $\mathcal{G} = \{p_i, q_j, r_k, \dots\}$ is a list of nodes and the sequence number of the message that was last delivered by each of these nodes.

**Current view** is the group view maintained at each node. Applications at the nodes are informed of changes to this current view. Applications at all nodes are assured that either they can communicate with all members of the current view, or they will be informed of a new current view.

**Tentative view** at a node includes all the nodes with which it can exchange messages. Apart from including nodes from the current view, the tentative view also includes nodes that are not yet members of the current view, but the broadcast layer is receiving and delivering messages from these nodes. Each node uses the list of suspected members available from the failure detector to exclude any suspected nodes from the tentative view. The tentative view at a node therefore maintains the view that the node expects to install next.

**Precedence ($<$) relationship** A group view $\mathcal{G}$ is said to "precede" $\mathcal{G}'$, written as $\mathcal{G} < \mathcal{G}'$, if $\mathcal{G}$ has at least one node with a last delivered sequence number lower than the one in $\mathcal{G}'$. Thus, view $\{p_i, q_j, r_k\}$ precedes the view $\{p_{i+1}, q_j, r_k\}$.

**Subset ($\subset$) Relationship** A group view $\mathcal{G}$ is said to be a subset of group view $\mathcal{G}'$, $\mathcal{G} \subset \mathcal{G}$, if $\mathcal{G}'$ includes at least one node that is not included in $\mathcal{G}$.

**List of Tentative view, *tviews*** Each node maintains a list of tentative views (*tviews*) that contains views awaiting agreement. The use of a list of next possible view allows TransMAN to handle multiple proposed group views in parallel and thus enable it to handle transient network conditions that are expected to occur frequently in MANETs.

### 4.5.1 Membership Agreement Protocol

The membership agreement protocol consists of two phases. The first phase determines a group view that can be the next group view, and the second phase runs an agreement protocol to finalise this next group view. The activity diagram in Figure 4.10 show the two phases of the membership agreement protocol and the events that result in the transitions in and out of the phases.

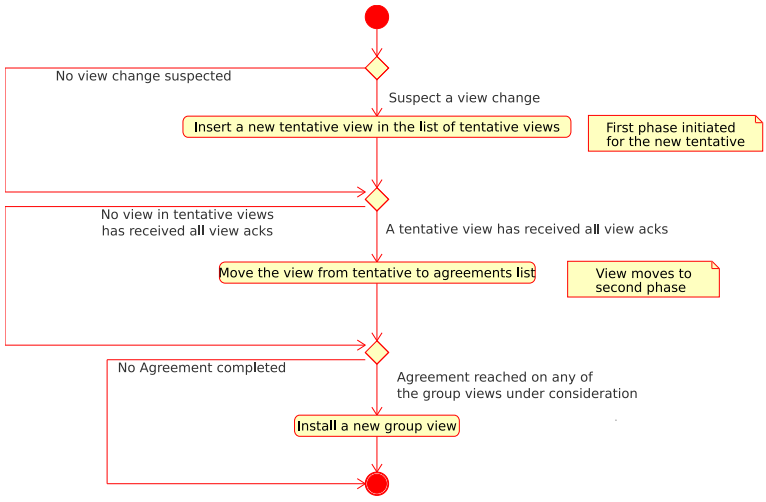


Figure 4.10: Activity Diagram - Two Phase Agreement

### First Phase: View Determination

The first phase of the membership agreement is initialised when a node realises the need for a view change. This can be caused by either (1) a node receiving a message from a non-member or (2) a node suspecting a member. To initiate the first phase, a node broadcasts a message containing its tentative view. This message is called the *init-view-change* message.

When a node receives an *init-view-change* message, it compares the tentative view that is included in the received *init-view-change* message to its current tentative view. Depending on the comparison between the two tentative views and the state of the membership agreement at the node, the node responds by either sending a new *init-view-change* message or updating the state of the membership agreement.

A node sends a new *init-view-change* message in response to the received one if the received tentative view has some information that the receiving node's tentative view does not have. This new information is either a higher sequence number of the message last delivered by a node or the presence of a new node. If the received *init-view-change* message proposes the same view as the one the received node maintains, then the receiving node updates the state of the membership agreement protocol (Table 4.1). These requirements are described by the first rule for the membership agreement is protocol.

**Rule** $\mathcal{AR}1$ – Let $\mathcal{T}$ denote the tentative view maintained at the receiving node and $\mathcal{T}_{rcvd}$ the tentative view received. A node's reaction to reception of $\mathcal{T}_{rcvd}$ depends on the relationship between $\mathcal{T}$ and $\mathcal{T}_{rcvd}$. The possible reactions are:

1. If $\mathcal{T} < \mathcal{T}_{rcvd}$ or $\mathcal{T} \subset \mathcal{T}_{rcvd}$, the receiving node updates $\mathcal{T}$ to $\mathcal{T}_{rcvd}$ and sends a new *init-view-change* message. Updating the view $\mathcal{T}$ to $\mathcal{T}_{rcvd}$ means updating the last delivered sequence numbers and incorporating the new nodes from $\mathcal{T}_{rcvd}$.
2. If $\mathcal{T} > \mathcal{T}_{rcvd}$ and $\mathcal{T} \not\subset \mathcal{T}_{rcvd}$ then the receiving node neither sends a new *init-view-change* message nor updates it tentative view.
3. If $\mathcal{T} \supset \mathcal{T}_{rcvd}$ then the $\mathcal{T}_{rcvd}$ is notifying the failure suspicion of a node at the sender. In this case the receiving node neither sends a new *init-view-change*

| View | view_acks | OPD values | | |
|---|---|---|---|---|
| | | $p$ | $q$ | $r$ |
| $p_i, q_j, r_k$ | $p_x$ | $OPD(p, p_x, mld[p])$ | $OPD(p, p_x, mld[q])$ | $OPD(p, p_x, mld[r])$ |
| | $q_y$ | $OPD(p, q_y, mld[p])$ | $OPD(p, q_y, mld[q])$ | $OPD(p, q_y, mld[r])$ |
| | $r_z$ | $OPD(p, r_z, mld[p])$ | $OPD(p, r_z, mld[q])$ | $OPD(p, r_z, mld[r])$ |

Table 4.1: Example membership agreement at node $p$

message nor updates it tentative view. Instead the receiving node waits until it suspects the failed node and then sends a new *init-view-change*. This avoids false suspicions resulting in generating extraneous membership agreements.

4. If $\mathcal{T} = \mathcal{T}_{rcvd}$, that is, the received tentative view is equal (both in nodes and their sequence numbers) to the one maintained at the node, the node treats the received *init-view-change* message as a *view-acknowledgement* for the node's tentative view. The node maintains a list of these *view-acknowledgements* received for each membership agreement.

Table 4.1 shows the data structure maintained by each node to track a membership agreement, *agr*. The group view under consideration is maintained in the leftmost column *agr.view*. The next column *view_acks* tracks the *view-acknowledgements* from the nodes in the view. These are written as *agr.view* and *agr.view_acks* for an agreement *agr*.

The first phase for a membership agreement at a node is completed when the node has delivered *view-acknowledgements* from all the nodes in the tentative view. This implies that all nodes propose the same group view as the next group view and is specified as the rule $\mathcal{AR}2$ of the agreement protocol.

**Rule** $\mathcal{AR}2$ – An agreement is run for a proposed group view only if all nodes in the group view propose the same group view.

If during the first phase any node detects a new node or a failed node, it sends an *init-view-change* with a new suggested view, resulting in the first phase being run again. If this new first phase completes, that is all nodes propose the same group view, then this view

moves to the second phase as well. There can be more than one agreement simultaneously being considered as next group view.

The use of lists of agreements in the first and second phases of the membership agreement enable TransMANs to handle transient view changes resulting from the transient network connections in MANETs. TransMAN initiates a new agreement every time there is a transient network change because if only one group view is considered as the next possible group view and the network conditions change again, the membership service will be tracking a group view change which should never be installed. Using a list of next possible group views avoids redundant or obsolete group view suggestions blocking other possible group view suggestions.

It is important to note that tracking more than one group view changes increases the computation costs at each node, but does not increase the communication cost. This is because each message helps make progress on all agreements. The details are discussed in the next section.

Module 8 on page 152 (Appendix C) shows the pseudocode for the first phase. The rule $\mathcal{AR}1$ is shown from lines 11 to 20 and lines 29 to 33. The transition from the first to the second phase is shown on lines 34 to 36.

#### Second Phase: Agreement Termination

In the second phase, an agreement for the tentative view is initialised and the tentative view is pushed into *tviews*, which is a list of views currently under consideration for next view. The completion of this agreement for the tentative view finishes the second phase. At the end, all nodes will agree on the same view as the next view.

A membership agreement for a group ($agr.view = \{p_i, q_j, r_k\}$) terminates at node $p$ when $p$ determines that all other nodes in $agr.view$ have delivered the *view-acknowledgements* sent by all the nodes in $agr.view$. These $agr.view\_acks$ were collected in the first phase of the protocol. Table 4.1 shows an example membership agreement run at node $p$, for a view $p_i, q_j, r_k$. The agreement shown is completed at $p$ when all the $OPD$ values shown in the table evaluate to true. This requirement is stated as the third rule for the agreement

protocol.

**Rule $\mathcal{AR}3$** – A node $p$ reaches agreement $agr$ on group view $agr.view$ if and only if

$$\forall n \in agr.view \quad \wedge \quad \forall va \in agr.view\_acks$$
$$OPD(n, va, mld[va.sender]) = true \tag{4.3}$$

**Theorem 13.** *If a node $p$ in $\mathcal{G}$ reaches agreement, $agr$, on group view, $agr.view = \mathcal{G}'$ and all nodes in $\mathcal{G}'$ remain reachable from each other, then all nodes in $\mathcal{G}'$ either reach the agreement $agr$ or initiate a new agreement on a successor group view.*

*Proof.* From rule $\mathcal{AR}3$, when $p$ reaches agreement on group view $agr.view$ it has determined all $va \in agr.view\_acks$ as stabilised. By the definition of message stability (Lemma 8) all nodes in $\mathcal{G}$ have delivered all $va$s. Therefore, if nodes in $\mathcal{G}'$ remain reachable they will eventually stabilise $va$s and will reach agreement on $agr$. However, if any node $q \in \mathcal{G}'$ becomes unreachable from nodes in $[\mathcal{G}' - q]$, these nodes will eventually suspect $q$ because of the eventual accuracy property of the failure detector. On detecting $q$'s failure, the nodes in $[\mathcal{G}' - q]$ initiate a new view change agreement. □

From the rule $\mathcal{AR}3$, the definitions of message stability and Lemma 8 the following lemmas are stated. These lemmas are later useful for proving the virtual synchrony property for TransMAN.

**Lemma 14.** *If a node $p$ reaches agreement, $agr$, on a group view $agr.view$ then all $va \in agr.view\_acks$ are eventually delivered at all nodes before they reach the agreement $agr$.*

**Lemma 15.** *If a node $p$ in group view $\mathcal{G}$ reaches agreement on $\mathcal{G}'$ then all nodes $n \in \mathcal{G} \cap \mathcal{G}'$ deliver a set of messages $M$ s.t. $\forall m \in M \wedge \forall va \in agr.view\_acks$, $m$ is a dependency of $va$, i.e., $m \rightarrow va$.*

The requirement that all nodes in a group view $\mathcal{G}$ determine that all other nodes have delivered the same set of $agr.view\_acks$ guarantees that all nodes agree on the same suggested view after delivering the same set of messages. The membership agreement

protocol does not tolerate any failures. If any nodes fail, the membership service initiates a new membership agreement from the first phase onwards.

Maintaining a list of tentative views for agreement in *tviews*, allows nodes to respond to multiple network events, i.e., node insertions or node failures. As a result, allowing a number of views may be considered for installation as next group views. This is desirable in a dynamic environment such as MANETs where frequent network changes are expected.

If an agreement for a tentative view does not terminate, the protocol ensures that it is deleted when a later[1] agreement terminates. The removal of agreements that are never reached removes any nodes that transiently connected to the group from any further consideration. However, if the transiently connected node reconnects to the group, a new membership agreement is initiated, where the new node is again considered for a group membership. The reliable broadcast continues to deliver messages during such transient connections and these message deliveries help in reaching agreement on group view changes. The Section 4.5.3 discusses how message stability is affected during such transient network connections.

No special messages are required to reach agreement. Instead, the application messages, or timeout messages in absence of application messages, are used to reach agreement. There is no traffic overhead in running a number of membership agreements. Instead, each message contributes towards the progress of all parallel agreements.

The last rule of the agreement protocol specifies that an agreement that includes a transiently connected node is deleted from the list of tentative views when a future agreement reaches termination.

**Rule $\mathcal{AR}4$** – When a node $p$ reaches agreement on view $\mathcal{T}$, $p$ deletes all $\mathcal{T}' \in tviews$ such that $\mathcal{T} < \mathcal{T}'$.

The second phase for an agreement can finish in two ways: (1) when an agreement protocol terminates or (2) when an agreement later in the agreement list terminates. In the latter case, the membership protocol deletes all agreements in the agreement list that

[1] later is defined in terms of the $<$ relationship.

precede the terminated agreement in the list of tentative views. If, during the second phase, a new suggested view is encountered that finishes the first phase, it is added to *tviews*.

The lines 22 to 27 of Module 8 on page 152 (Appendix C) and the procedure UPDATEAGREEMENT of Module 9 on page 153 show how the second phase of an agreement is updated on message delivery. Once agreement is reached, the new view is installed, as shown in procedure INSTALLVIEW of Module 9. Line 17 shows that agreements are terminated if some future agreement is reached.

### State Machine For The Agreement Protocol

The state transition diagram in Figure 4.11 shows the states for the membership protocol. The STABLE state implies a view is installed, no changes in the view are suggested and no agreements are in the second phase. On receiving a message from a member node the state remains STABLE and the message is delivered and stabilised as per the reliable broadcast and stabilisation protocols.

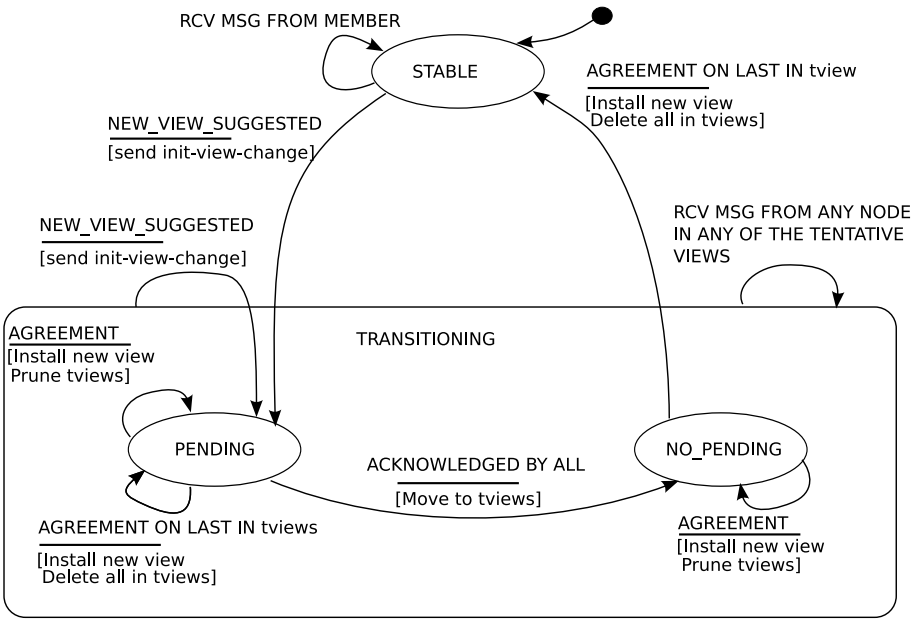


Figure 4.11: State Transition Diagram for Membership Protocol

If a node receives a message from a new node, or suspects a node in its view or it receives

an *init-view-change* message, the node initiates the view change protocol. In the state transition diagram, these events are encapsulated by the NEW_VIEW_SUGGESTED transition. On this transition, the protocol moves from the STABLE to the TRANSITIONING state.

The TRANSITIONING state has two sub-states, PENDING and NO_PENDING. These correspond to the presence or absence of an agreement in the first phase. The protocol is in the PENDING state if there is at least one agreement in the first phase. The protocol moves to NO_PENDING when all agreements for a tentative view have moved to the second phase. At this point there is no agreement in the first phase and the state is called NO_PENDING. If a NEW_VIEW_SUGGESTED event occurs in the NO_PENDING state, the protocol moves back to the PENDING state.

### Handling MANET's Dynamic Environment

Figure 4.12 shows an example run where nodes $p$ and $q$ do not fail, but $r$ fails and a new node $s$ joins the run. The Figure shows the states the membership agreement protocol goes through. The states are shown only when all nodes have reached the same state, although it is not necessary that all nodes go through the same states. The Figure shows how the views $\{p, q\}$ and $\{p, q, s\}$ both finish the first phase and await the termination of the corresponding membership agreements. The membership protocol installs the view $\{p, q, s\}$ and deletes $\{p, q\}$ from the agreement list.

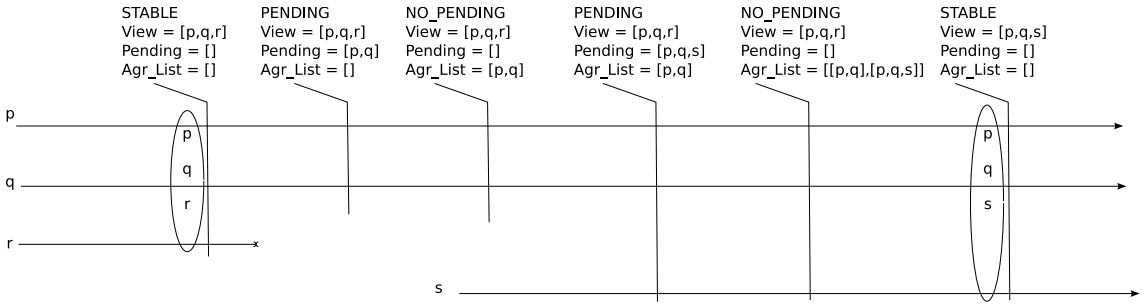


Figure 4.12: Example State Transitions for the Membership Protocol

The protocol moves back to the STABLE state when the last agreement in the *tviews*

is terminated while the protocol is in the NO_PENDING state. In contrast, if in the NO_PENDING state an agreement is reached which is not the last one in the *tviews*, the protocol stays in the TRANSITIONING state.

### 4.5.2 Virtual Synchrony and Transitional Sets

The algorithms described in the previous sections support FIFO and total ordered message delivery. The protocols also provide message delivery such that the set of nodes that survive a view change deliver the same set of messages. This property is called "virtual synchrony" and this section explains its provision in TransMAN.

Suppose a new view $\mathcal{G}$ is proposed, such that it is a merger of two partitions $\mathcal{G}'$ and $\mathcal{G}''$. The membership agreement ensures that all nodes in the proposed view $\mathcal{G}$ have delivered at least all messages that are dependencies of the $agr.view\_acks$. This requires that nodes in $\mathcal{G}'$ and $\mathcal{G}''$ deliver the same set of messages between them, before they install $\mathcal{G}$. Once the proposed view is installed, all nodes deliver the same set of future messages. All nodes that survive a view change deliver the same set of messages.

The approach to delivering and stabilising messages and to sending the current group view with every *init-view-change* message, allows a node to determine if the principle of virtual synchrony holds. This is achieved by calculating the transitional set (Moser, Amir, Melliar-Smith & Agarwal 1994) at every view installation. The transitional set is easy to determine as described in (Chockler et al. 2001) using the current view sent by each node with the *init-view-change* message.

### 4.5.3 Message Delivery During View Changes

This section describes how reliable message delivery and stabilisation are affected when a MANET changes such that the membership agreement protocol is initiated. Three different network and node events are considered, namely a merge between groups, a member failure and the false suspicion of a member. These conditions are considered here because they are expected to occur quite frequently in a MANET and because TransMAN claims to handle these situations without forcing applications to block from sending messages.

1. *Merge:* The reliable broadcast protocol utilised by TransMAN allows delivery of messages from nodes that are not part of the current view. This supports FIFO delivery of messages to all nodes from all participating nodes.

2. *Failure:* If a node fails or the network partitions, then messages are not stabilised because the failed or partitioned group members are unable to acknowledge message delivery. On the other hand, the reliability layer continues to deliver messages, which allows FIFO delivery of messages among the surviving nodes.

3. *False Suspicion:* In the case of false suspicions, a group view agreement is initiated but all messages are delivered and stabilised as governed by the reliable broadcast and stability protocols. This is possible because all members of the group continue to broadcast messages even if a node is falsely suspected. These members will eventually implicitly acknowledge all messages received from the falsely suspected node.

## 4.6 Group Communication Properties

The Table 3.2 on page 57 showed the mapping of TransMAN's high level goals to the specification listed in Appendix A. This section shows how the protocols described in this chapter meet the high level requirements of (1) Reliable Broadcast, (2) Ordered Delivery, (3) Consistent List of Participating Nodes, (4) Non-Blocking View Changes and (5) Virtual Synchrony. Each of these system requirements are considered and proofs are presented for the properties that together help implement the requirement.

**Reliable Broadcast and Ordered Delivery** are provided by the properties *Delivery Integrity*(A.8), *No Duplication*(A.9), *Self Delivery*(A.10), *Reliable FIFO*(A.14) and *Weak Total Order*(A.15 and A.16).

*Delivery Integrity* is satisfied by the no spurious message assumption that constrains messages in the system to have originated only from nodes that are participating the GCS. Any received message that does not confirm the TransMAN header specification in Figure 4.3 is discarded by the system.

*No Duplication* is satisfied because messages are only considered once for delivery, as shown in line 6 of Module 3 (Appendix C). If a duplicate is received then it is considered delivered only if it is of type of "init-view-change", as per rule $\mathcal{AR}1$. This is only for the purposes of sending *view_acks* in response to receiving "init-view-change" messages and these duplicates are not delivered to applications using TransMAN.

*Self Delivery* for *Reliable FIFO* follows directly from Theorems 4, 5 and 6 that state that if nodes remain reachable from each other they eventually receive all messages sent by all the nodes. All messages are delivered by all nodes as and when the dependencies of the messages are received. *Weak Total order* is proved as Theorem 12.

**Consistent List of Participating Nodes and Non Blocking View Changes** are together provided by the *View Coherency*(A.6)(A.7) , *Local Monotonicity*(A.2), *Self Inclusion*(A.1), *Initial View Event*(A.3), *Membership Accuracy*(A.4) and *Termination of Delivery*(A.5).

*View Coherency* is defined in two parts, shown in Equation (A.6) and (A.7). The first part requires that if a process installs a view $\mathcal{G}$ then either all nodes in $\mathcal{G}$ install $\mathcal{G}$ or $\mathcal{G}' > \mathcal{G}$, and is proved as Theorem 13. The second part requires that if two nodes ($p$ and $q$) install the same view $\mathcal{G}$ and if one of them installs a new view $\mathcal{G}'$, then the other node installs the new view as well or it crashes. This follows from the *Strong Completeness* property of the failure detector, which is the use of *failure to receive* for failure detection and the *Eventual Reception* property of the broadcast protocol. A node $q$ does not install the new view $\mathcal{G}$ only if $q$ is not receiving messages from other nodes. The failure detectors at the other nodes observe that $q$ is *failing to receive* messages and start suspecting $q$ as crashed.

*Local Monotonicity* is guaranteed by the use of sequence number on messages. Every message sent by a node is a recognisable successor to the previous message. *Self Inclusion* follows from the rule $\mathcal{AR}3$ of the membership agreement protocol that

requires a node to propose a view based on its tentative view, which by definition includes the sending node. *Initial View Event* is guaranteed because each starts with an initial group view that includes itself, and each message is therefore sent in some group view.

*Membership Accuracy* is guaranteed from the *Eventual Reception* and rule $\mathcal{AR}1$ of the agreement protocol. Eventual reception property guarantees that if nodes remain reachable from each other they eventually receive messages from each other. Following a message reception from a new node, $\mathcal{AR}1$ initiates a new membership agreement protocol and Theorem 13 guarantees the termination of the agreement to either install the new proposed view or some later group view that includes the new nodes (if they are still reachable).

*Termination of Delivery* requires that if some node $p$ in a group view stop delivering messages from other nodes in the view then the other nodes install a new view excluding $p$ from the group. The property is guaranteed because of the use of *Failure to receive* by the failure detector. When nodes realise that $p$ is not receiving messages from them they start suspecting it and initiate a new membership agreement protocol. This terminates in the installation of a group view that excludes $p$, if $p$ remains unreachable from the rest of the group.

**Virtual Synchrony** requires that nodes that survive the same group view changes deliver the same set of messages. Lemmas 14 and 15 directly prove TransMAN's support for virtual synchrony.

## 4.7 Conclusions

The chapter presented the design of the TransMAN group communication system, which includes: a reliable broadcast service that can use existing optimisation; a message stability protocol; a failure detector; and a membership agreement protocol that supports virtual synchronous group view changes without blocking applications from sending messages during the view changes. The design presented is motivated to tackle the challenges presented

by MANETs including frequent topology changes, lack of collision detection in IEEE's 802.11b MAC, limited battery lives of mobile devices and transient network conditions.

The central concept of the design is the use of message dependencies and maintaining a "shared message space" in a graph at each node. The graph at each node is used to determine message stability and provide agreement on membership view changes. The use of this graph adds a computational overhead for every message received by a node, but is chosen to reduce the number of message transmissions when compared to schemes that use explicit positive acknowledgements and agreement control messages.

The agreement protocol presented in the chapter allows a number of views to be considered as possible next group views. The approach is chosen to handle the frequent transient changes in network connectivity expected in MANETs. Every time the network connectivity changes a new membership agreement is proposed. All these agreements are tracked using the same messages, i.e., at the same transmission cost as that of maintaining one agreement.

The design presented here is implemented on an open source platform and the source is made available to allow independent future improvements and comparative evaluations of the design presented.

# Chapter 5

# Implementation

The protocols described in the previous chapter are implemented for a real world evaluation and this chapter describes the implementation details. The implementation is available for GNU systems and uses open source libraries to facilitate future independent improvements and comparative evaluations. A real world implementation is chosen over a simulation because of the contemporary controversy surrounding results from simulators (Cahen, Sasson & Schiper 2002, Andel & Yasinsac 2006, Kurkowski, Camp & Colagrosso 2005) and also because system implementations are more conducive to comparative evaluations. The lack of an implementation for a MANET GCS further motivates the choice of a real world implementation.

## 5.1 Using TransMAN

TransMAN is implemented as a stand-alone daemon and applications interact with it using UDP socket IPC. Applications use ruby libraries supplied with TransMAN to construct messages and marshal them for communicating with the TransMAN daemon. The TransMAN daemon then appends its header to the message and broadcasts it to the group. Figure 5.1 shows the interprocess communication between the application and the TransMAN daemon (the heavy lines show the IPC calls).

TransMAN differentiates between application messages and timeout messages it gener-

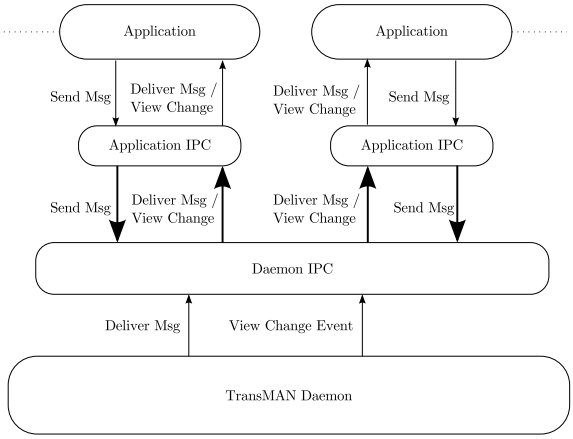


Figure 5.1: Application Communication With TransMAN Daemon

ates by associating a message type to each message sent. The structure of the application message sent to the TransMAN daemon is shown in the Figure 5.2. TransMAN demultiplexes application messages between application by using the "app-id" field of the message delivered.

| app id | seq no | from | type | data |
|---|---|---|---|---|

Figure 5.2: The Structure of the Application Message Handed to TransMAN

The component "Daemon IPC" demultiplexes messages to applications that are using the TransMAN daemon. Allowing a number of application to utilise the same TransMAN daemon allows TransMAN to track group membership on a per node basis instead of a per application basis. This implies that for the TransMAN daemon an ad-hoc network is composed of a single group and applications can attach additional filters to associate themselves to particular groups that are sub-groups of the MANET group.

The IPC between the application and the TransMAN daemon is implemented using UNIX domain sockets. This bypasses the network stack because it is assumed that the application and the TransMAN daemon are co-hosted on the same node.

### 5.1.1 The Issue of Group Size

Support for a MANET-wide group raises questions about TransMAN's scalability to a large ad hoc network. Gupta & Kumar (2000) point out limitations on the size of ad hoc networks (using IEEE 802.11b) that directly affect the maximum size of a group in a MANET. Such questions point towards an upper bound on the size of a MANET and other researchers (Boppana & Zheng 2005) argue for the case of MANETs of up to 1000 and more nodes. These studies focussed on a large MANET are concerned with either unreliable or unicast communication and not with a many-to-many reliable communication.

From existing literature (Birman et al. 2001, Keidar et al. 2002, Birman et al. 1999, Das et al. 2002) on group communication systems for LANs and WANs, it is clear that group communication is found practical for networks of sizes of the order of tens of nodes. Das et al. (2002) argue that providing strong reliability guarantees for groups of the order of hundreds of nodes is only feasible by compromising on the reliability guarantees and show that gossip-based techniques (Birman et al. 1999) are best suited for groups of such large sizes.

The two issues of the maximum size of a MANET and the compromise of reliability guarantees for large group sizes place strong limitations on the group size that can be supported in a MANET. The TransMAN implementation does not provide for means to limit the size of a group, raising the question of scalability. This question is answered by architecturing TransMAN such that it supports the use of best available broadcast protocol in a MANET. TransMAN scales as well as the underlying broadcast protocol, given the extra processing required to maintain membership information is available at each of the nodes.

Limitations on the size of the MANET or a group can be imposed by using extra information like the location of nodes. Such optimisations and extensions are not discussed in this thesis but are easily applicable if such location services are available.

### 5.1.2 Broadcast Optimisation Used

A flooding scheme can be used to support the TransMAN protocols, just like any other optimised broadcast scheme. However, a simple scheme to reduce redundant transmissions is used to improve the responsiveness and message delivery latencies. The scheme used is the counter-based optimisation suggested by Tseng et al. (2002). This optimisation is implemented as a strategy class (Gamma, Helm, Johnson & Vlissides 1995, Chapter 5) so that any improved optimisation can easily replace the current optimisation.

## 5.2 Environment

TransMAN is currently implemented for GNU systems[1] using the Ruby programming language[2]. The choice of the environment is made to ease rapid development. The current implementation is made available under the BSD license[3] on RubyForge[4].

The implementation uses Ruby's libraries for graph manipulations and threads. All the libraries used are open source and are readily available on numerous platform. This allows independent improvements and future comparative evaluations with the TransMAN implementation.

## 5.3 Implementation Design

This section describes the implementation architecture for TransMAN. The class diagrams for the main classes are presented along with communication diagrams showing the important interactions between the implementation objects. The source code for the implementation is available from RubyForge.

The Figure 5.3 shows the main classes used in the implementation. The class *ReliabilityAgent* is the main interface of the system to the lower communication layers: *SendAgent* and *ReceiveAgent*. These lower communication layers communicate with *ReliabilityAgent*

[1] http://www.gnu.org/
[2] http://www.ruby-lang.org/
[3] http://www.opensource.org/licenses/bsd-license.php
[4] http://rubyforge.org/projects/transman/

using thread safe queues available in the Ruby core[5]. *ReliabilityAgent* maintains references to instances of the *DBG*, *Stability*, *FailureDetector* and *Membership* classes.

The *Membership* class has a subclass *StaticMembership* that is used only for evaluation purposes by providing a dummy class maintaining references to a static group view. This static view known at the outset and does not change during a system run. *Membership* includes instances of *GroupView* that maintain the current group and the list of suspected nodes. A *GroupView* holds a list of [*node_id*, *mid*] pair, where the *mid* is the message last delivered from the *node_id*. Including the *mid* in the *GroupView* allows the node to distinguish two different group views with the same list of nodes, as discussed in Section 4.5 on page 77.

*Membership* maintains a list of *MembershipAgreement*s, each of which includes a group view as the "proposed" group view and a list of *mid*s that represent the "view-acknowledgements" received from the nodes in the proposed group view. *MemebershipAgreement* supports methods that allow membership and reliability objects to update the agreements on delivery of every message and to check if any agreements have terminated.

Figure 5.4 shows the *IPCMessage* header used for communication between between the TransMAN daemon and the application. The figure also shows the *BcastMessage* used for communication between the wireless nodes. Each instance of a *BcastMesssage* includes a header field and a data field. The header field of the *BcastMessage* includes a header which contains the *mid* and the dependencies of the message. These dependencies are the central part of TransMAN and are used to provide message reliability, stability and membership agreement.

The Figure 5.5 shows the UML communication diagram for message reception at the *ReliabilityAgent* from the *ReceiveAgent*. The Figure shows the case when the received message can not be delivered and is inserted in the list of *received_messages*. A negative acknowledgement is sent for the missing dependency that prevented this message from being delivered. *SendAgent* hands over this request of sending a negative acknowledgement to the *CounterBasedOptimiser* that handles all message receptions and transmissions. The

[5] http://www.ruby-doc.org/core/

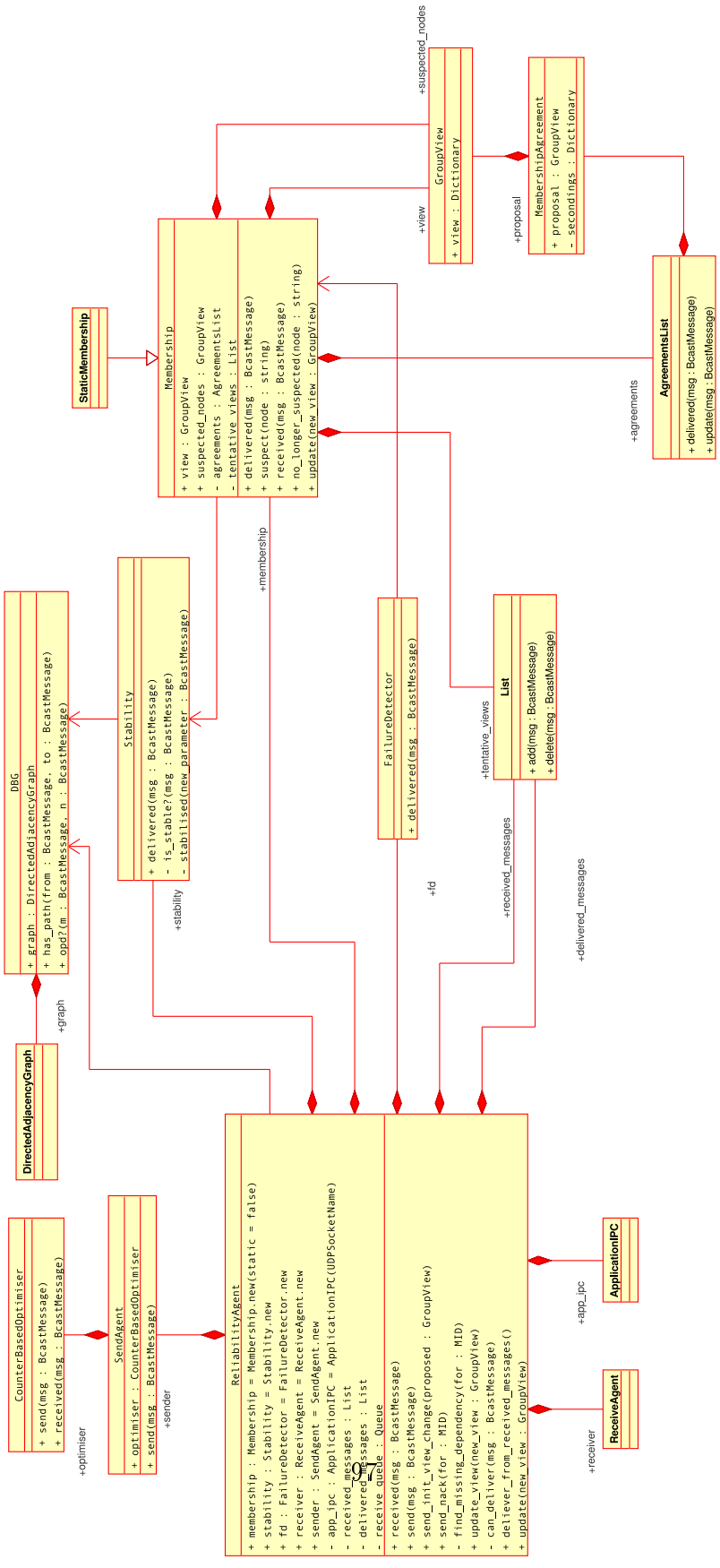


Figure 5.3: UML Class Diagram For TransMAN

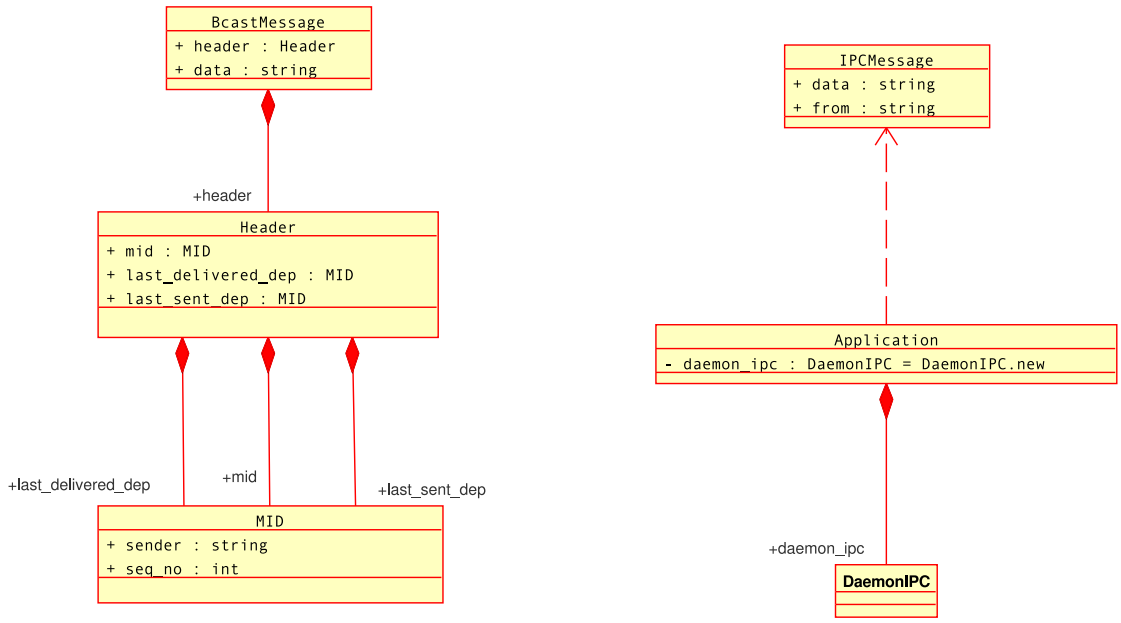


Figure 5.4: Classes Representing Message Headers

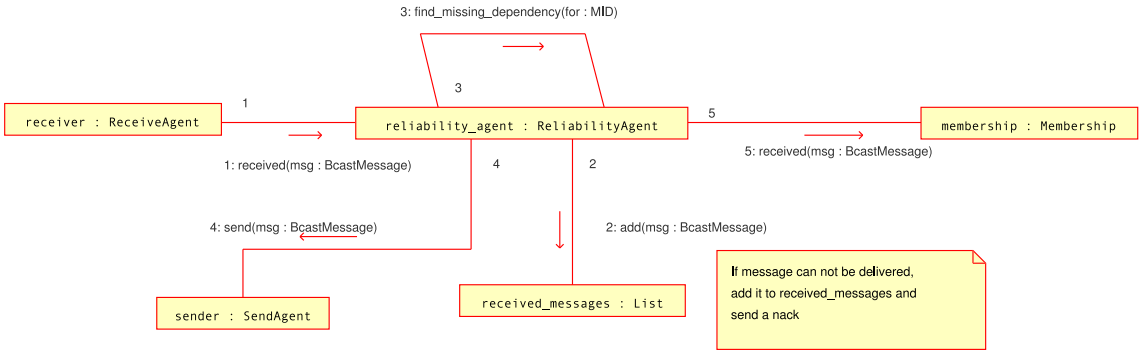


Figure 5.5: Communication Diagram for Message Reception

counter-based optimisation can be replaced with another broadcast optimiser to provide a better performance by simply replacing the *CounterBasedOptimiser* class with another. The response of the membership layer on receiving a message from the *ReliabilityAgent* is shown in Figure 5.7 and discussed on the following page.

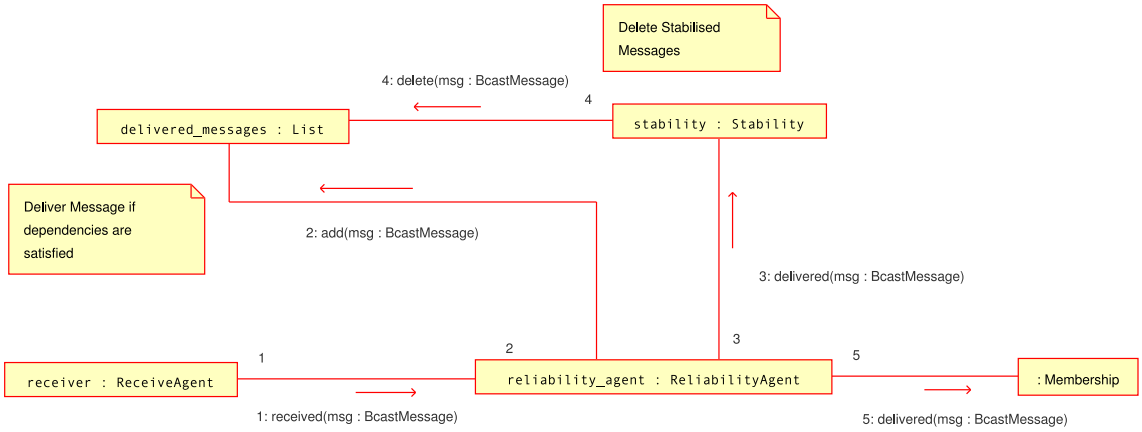


Figure 5.6: Communication Diagram for Message Delivery

Figure 5.6 shows the communication between objects to handle a message reception when all the dependencies of the message have been received and the message can be delivered. On determining that the message is deliverable, the *ReliabilityAgent* inserts the message in the list of *delivered_messages* and notifies the *Stability* object of the message delivery. The *Stability* object then checks for any messages that are stabilised because of the current message delivery. If there are any messages determined as stable, the *stability* object deletes the stabilised message from the list of *delivered_messages*. It also cleans up the *DBG* to delete the stabilised messages that are no longer required for any membership agreement. Pruning of the data structures is essential for not only reduced memory utilisation, but also for reduced processing time when a message is delivered. Pruning *delivered_messages* is important because whenever a message is delivered, all messages in the list are checked for stability and each check for message stability is an expensive operation involving checking for paths between nodes in the *DBG*, as explained in Section 4.3.2.

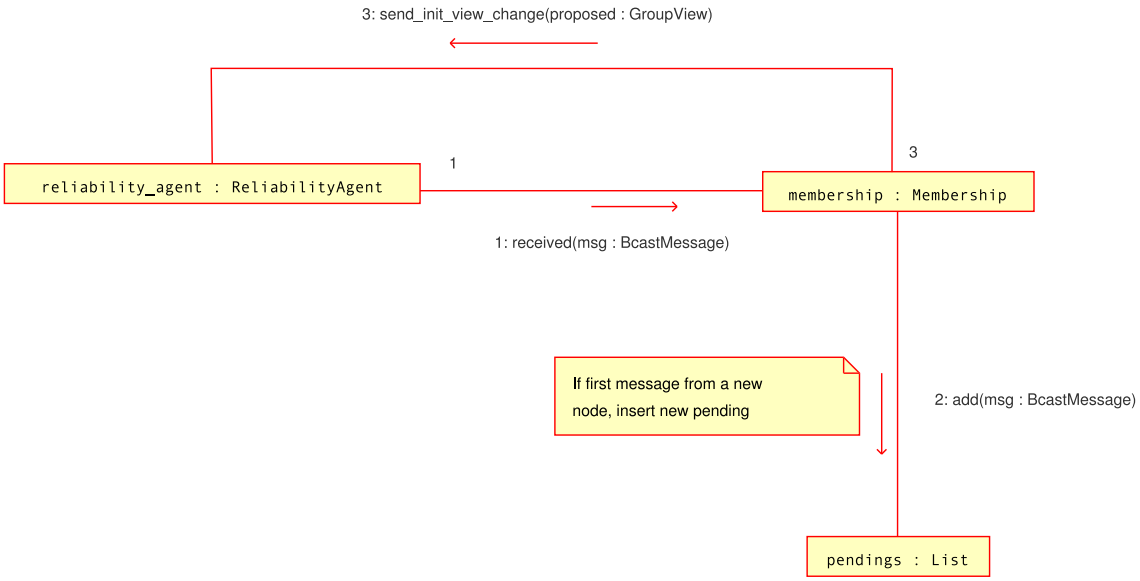


Figure 5.7: Communication Diagram for Membership: Receiving New Message

Figure 5.7 shows the communication between objects when a message is received and the *Membership* object is informed of the event (see also Figure 5.5). If the received message is the first message to be received from the sender, the *Membership* object initiates a new membership agreement protocol. The protocol is initiated in to the first phase (see Section 4.5.1) by changing the list of *pending* suggested views to reflect the presence of the new node. After updating the *pendings* list, an "init-view-change" message is broadcast by invoking a message on the *ReliabilityAgent*. The second phase of the agreement starts once view-acknowledgements are received from all the nodes in the suggested view. These view-acknowledgements are only noted when messages are delivered from the nodes in a suggested view.

Figure 5.8 shows how the *Membership* and *AgreementsList* interact to implement the second phase of the membership agreement protocol (see Section 4.5.1). Whenever a message is delivered by the *ReliabilityAgent*, the *Membership* object is informed of the event. The *Membership* object updates all the views in the list of suggested views about the delivery of the message. These suggested views update their view-acknowledgements and all the views that have now been seconded by all the nodes in the view are inserted

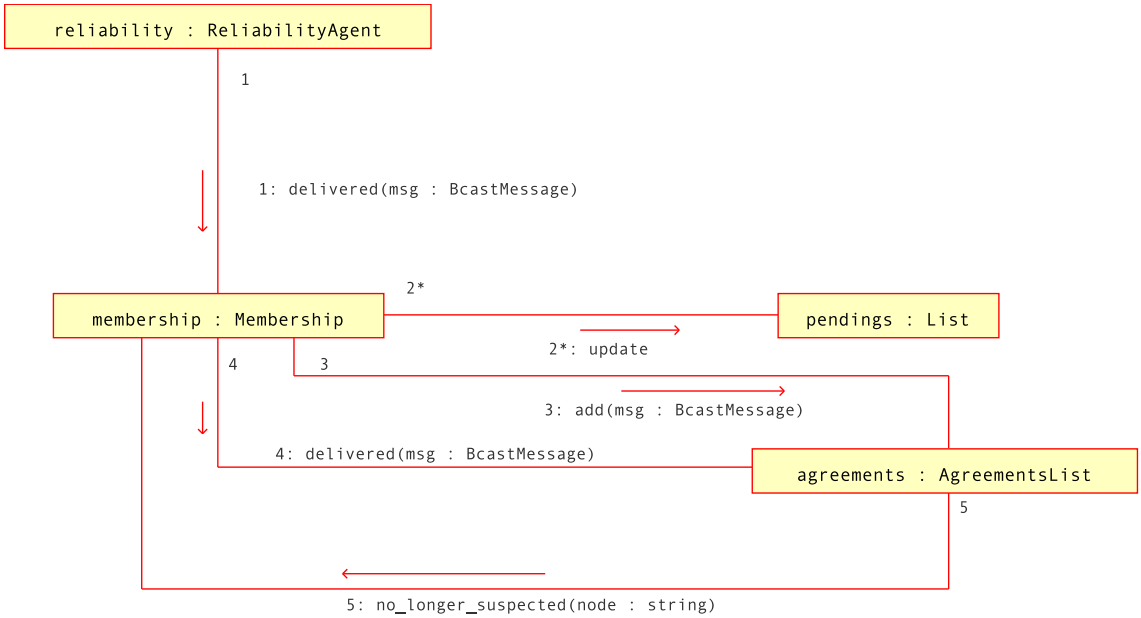


Figure 5.8: Communication Diagram for Membership: Delivering Messages

into the list of membership agreements, *AgreementsList*.

After updating the list of suggested views, the *Membership* object informs the list of tentative agreements about the delivery of the message. These agreements update their state table (shown in Table 4.1), which are stored as instances of the *MembershipAgreement*. Finally, if any *MembershipAgreement* has terminated the *ReliabilityAgent* is updated, which further informs any interested application of the view change via the *ApplicationIPC*.

## 5.4 Implementation Optimisations

This section presents the optimisations identified while conducting the evaluation study presented in the next chapter. Some of these optimisations are left as future work to improve the performance of the implementation.

### Checking for Stability

As the depth of the DBG increases, the processing time for checking message stability increases. This is because the DBG has to be traversed from the root to the leaves for determining message stability. Also an increase in the number of nodes in a group increases the width of the graph, this further increases the time to traverse the graph, increasing the processing times for determining message stability. The processing time for determining message stability therefore is proportional to both the group size and the depth of the DBG.

To avoid this processing time to grow exponentially the implementation prunes the DBG every time a message is stabilised. That is, whenever a message is stabilised it is deleted from the DBG if it is not required for any of the pending membership agreements.

This leads to interesting situations when the group size is large and a node fails or leaves the group. In such a situation the nodes remaining in the group initiate a group view change protocol and can not stabilise messages because of the inability to determine if the "suspected node" has delivered the message, see Section 4.5.3. This leads to a rapid increase in the depth of DBG with each delivered message, and the effect gets more influential with the increase in the size of the network (in turn the width of the DBG). As an optimisation the stability of messages is not checked while some nodes are suspected, this is in confirmation with the protocol behaviour during group view changes (see Section 4.5.3).

### Negative Acknowledgements

Whenever a message is received that can not be delivered, because some of its dependencies are not satisfied, a negative acknowledgement is sent for a message or one of its missing dependencies. Sometimes messages are received such that duplicates of a negative acknowledgement are sent for the same missing dependency.

This approach can be optimised so that if a nack for a message is waiting for a transmission, that is, it is present in the *send queue* of the sender agent, a duplicate of the nack is not inserted in the send queue. In the current implementation duplicate nacks are sent

as required by the protocol, that is duplicate nacks are not avoided.

While checking for missing dependencies of a received message the *received messages* list is checked, the *received queue* is not checked for the missing dependency. While this leads to a correct behaviour of the reliability protocol, it results in sending nacks for messages that are present in the receive agent's *received queue*. This approach in the current implementation can be optimised so that the *received queue* is checked for a missing dependency before a nack is sent for the same.

## 5.5 Conclusions

The chapter presented the design of the implementation using UML class and communication diagrams. The chapter also showed how an application can use the implementation by using an IPC mechanism to communicate with the TransMAN daemon. This chapter also explained the practicalities of the TransMAN implementation that will be evaluated in the next chapter.

The issue of scale is the most important in an ad hoc network. Different schools of thought exist on how large an ad hoc network can be, both in terms of the number of hops between the nodes and the number of nodes in a single hop. The real world implementation described in the thesis can be used in future studies to answer the question about the scale of the network. The next chapter evaluating the TransMAN implementation explores the problems of running a real world ad hoc network to support all to all communication.

# Chapter 6

# Evaluation

TransMAN provides reliable and ordered message delivery, a consistent list of participating nodes and virtually synchronous communication between mobile nodes using wireless communication. The guarantees are provided without requiring applications to block during view changes. Chapter 4 presented proofs that TransMAN supports these properties and while this chapter briefly comments on how this was verified, the main focus is on the performance overheads of providing these properties. TransMAN is designed to reduce the overhead resulting from the membership protocol, and an evaluation study that measures this overhead is presented.

The overhead of the TransMAN group membership service is measured as compared to the reliable broadcast protocol. During the study, the characteristics of the reliable broadcast protocol are observed when subject to the conditions created by the experimental setups. The behaviour of the system is studied while subjecting it to transient changes in network connectivity. Each experiment is run first for a network with no changes in network connectivity, and then with transient changes, observing any increase in overheads. To generate transient changes in network connectivity, the transmission power used by the nodes is adjusted at random time intervals.

## 6.1 Verifying Implementation of Functional Properties

The functional properties of reliable and ordered message delivery (FIFO and Total ordered) along with virtual synchrony is verified for each experiment run by running verification scripts on the logs generated by each run. These verification scripts are made available as part of the implementation source code available under BSD license.

The verification scripts parse the log files generated by the experiment runs and verify that (1) messages are delivered in sender FIFO order, (2) messages are stabilised in a Total order across all nodes, (3) nodes that survive the same group view changes deliver the same set of messages (virtual synchrony), and that (4) applications or timeout messages are FIFO delivered even as group view changes are taking place (non-blocking view changes). The scripts are available under the *thesis-evaluation* release files available from RubyForge[1].

## 6.2 Experimental Setup

The implementation is carried out on a GNU-based system and is made available under the BSD license from RubyForge. The choice of platform is motivated to encourage future enhancements to the implementation and independent comparative evaluations.

The experiments are run using ten laptops of various specifications, and up to eight random laptops are randomly chosen for each experiment run. This eliminates the bias towards any one topology or subset of laptops available. The specifications for the laptops is listed in the Table 6.1, which shows the variations in the processors used to run the experiment. The variations in the hardware used confirms the need for the assumptions about asynchronous networks with varying processor speeds and clock drifts (see Section 3.2).

The experimental setup uses Cisco (Aironet 350) 802.11b wireless cards without the external rabbit-ear antennas, as shown in Figure 6.1. The antennas were removed to reduce the range of transmission of the wireless signal originating from each node. Without the external antenna, but with the connecting wire, the transmission range of the signal

[1] http://rubyforge.org/projects/transman/

| **Number** | **Laptop Model** | **Processor** | **RAM(MB)** |
|---|---|---|---|
| 4 | Dell C400 | Intel Pentium III | 256 |
| 3 | Dell D400 | Intel M | 256 |
| 3 | Dell D410 | Intel Centrino | 512 |

Table 6.1: Specifications For Hardware Used

Figure 6.1: Antennae Without Rabbit Ears

is reduced to about two feet. This reduced transmission range is useful for conducting experiments and generating transient network conditions in a lab environment. With the rabbit ears, the range of each of the nodes is 250m and transient changes to network connectivity can not be generated as well as with reduced transmission ranges.

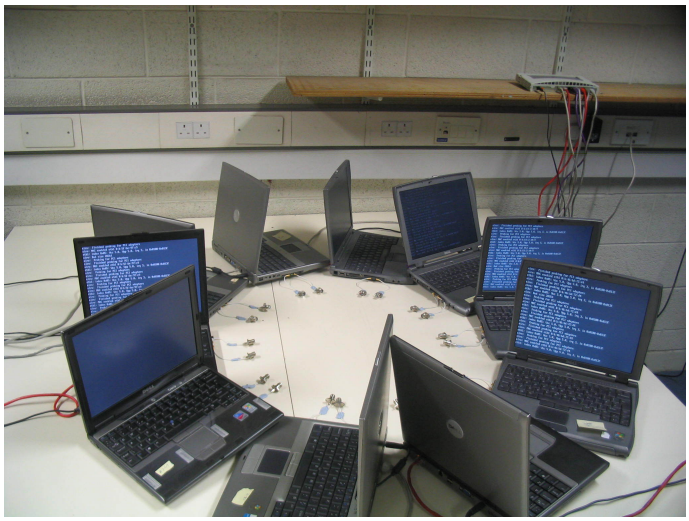

Figure 6.2: Strongly Connected Network of Laptops

The nodes in the network are setup in a circle to provide a strongly connected topology, where all nodes are connected to all other nodes via the wireless network interface (Figure 6.2). This allows a random subset of the laptops to be chosen for running experiments with different group sizes. All these subsets form strongly connected networks. By randomly choosing different laptops to run repetitions of an experiment for a fixed group size, the effect of the relative position of the laptops and other interferences is eliminated.

The signal strength without the rabbit ears is reduced to an extent that laptops can not communicate if they are at different elevations. This was discovered while setting up the experiments in a circle which extended to more than one table. During such an experimental setup, laptops were initially placed on two tables with a difference in height from the floor of about two centimetres. The tables had to be adjusted so that they were of the same height, otherwise the laptops on either of the tables were unable to communicate with the laptops on the other table. It is usually expected that a wireless signal propagates in all three dimensions and that the difference of two centimetres in

the heights of the laptops should not affect the communication between the laptops. The inability of the laptops to communicate while at different elevations reflects how low the signal strength is when the rabbit ears are not attached to the wireless cards.

To change the connectivity between nodes during an experiment, the transmission power used by a node's wireless card is adjusted. This determines how far the signal propagates, and can be adjusted at run time using the wireless configuration tools available with GNU systems[2]. To change the network connectivity between nodes at run time half the nodes in an experiment change the transmission power at random intervals between 10 and 20 seconds. Whenever a laptop changes its transmission power, it chooses a random value from those allowed by the Cisco (Aironet 350) card, i.e. 5, 20, 30 or 50mW.

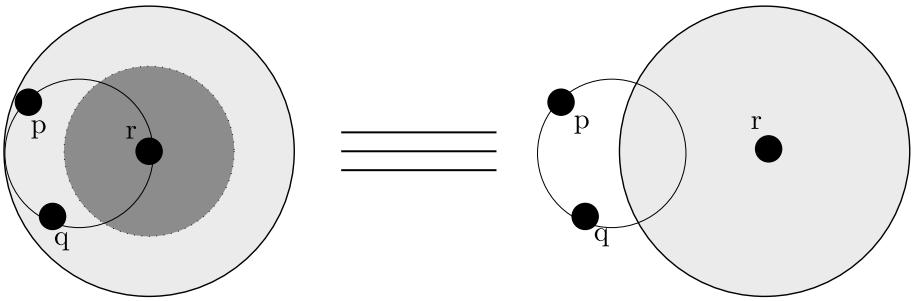


Figure 6.3: Radial Movement of Nodes

A change in transmission power simulates a node moving in a radial direction away from the network circle. To illustrate the effect, Figure 6.3 shows nodes $p, q$ and $r$ in a circle and node $r$'s transmission power is reduced (left of the figure). The effect is that nodes $p$ and $q$ can no longer receive messages from node $r$. This is similar to the situation where $r$ moves away from the circle in a radial direction (right of the figure).

The two situations, however, are not exactly the same. By reducing the transmission power of node $r$, only a unidirectional disconnection is possible. Even while $p$ and $q$ can not receive message from $r$, $r$ can still receive messages from $p$ and $q$, because their transmission ranges are unchanged and $r$ physically is still in their range. However, the scheme of changing the transmission power allows repeatable experiments to be set up in a lab environment while providing transient changes in network connectivity.

[2]The *iwconfig* utility supports adjusting the transmission power used by a wireless card.

Parameters that are not studied or whose repeatability can not be guaranteed are factored out by use of randomisation. For example, using random changes in transmission power allows the effect of node mobility to be studied without studying just one mobility scenario. Similarly other system variables like heartbeat are randomised (within a range) to eliminate the effect of any one value of such parameters. The laptops for each run of an experiment are also randomly chosen from a set of ten laptops, eliminating the effect of any particular hardware or the location of the laptops in an experiment.

The experiments are conducted in a lab environment where a number of other networks are running on all three frequency bands used by the IEEE 802.11b protocol. Experiments are run on channel six, chosen because it showed the least activity in the lab. The results shown in the above tables can not be be interpreted without correlating the network usage in the building. However, after running each of the experiments for 20 iterations and eliminating outliers from the observations, the affect of the other networks is expected to be minimal.

## 6.3 Parameters

The execution of the TransMAN protocol is parameterised by the heartbeat, waitlength and the counter value used for the counter based optimisation. The parameters and their importance to the experiments are described here.

### 6.3.1 Heartbeat

Heartbeat is the time period elapsed between two consecutive timeout messages sent by the TransMAN daemon. It affects delivery latencies of the messages and the reactivity of the group membership service. The larger the values of heartbeat the longer the delivery latency for messages. This is because the reliable broadcast protocol uses the future reception of messages to determine past lost messages. Membership reactivity is slowed down as the heartbeat values used become larger. This is because the agreement protocol uses reception of broadcast messages to reach agreement. This motivates the use of small heart-

beat values. However too many messages sent in a short interval will flood the network resulting in message losses and node starvation (Singh 2006).

Finding a suitable range of heartbeat for different group sizes is important, and so, a simple experiment is set up to determine the values to use in experiments with various group sizes. The experiment observes the delivery latency and the number of collisions for a network of two, four and eight nodes with varying values of heartbeat. For this experiment, only the reliable broadcast and message stability protocols are used. The membership protocol is avoided because the purpose of the experiment is to observe the effect of the heartbeat on the traffic and the delivery latencies in the network, and not to measure the influences of the membership service.

Earlier studies (Singh, Nedos, Gaertner & Clarke 2005) on the performance of the reliable broadcast and message stability protocols described in this thesis were conducted with heartbeat values in the range of 100-150 milliseconds. These earlier experiments were conducted using the network simulator ns-2 (Breslau, Estrin, Fall, Floyd, Heidemann, Helmy, Huang, McCanne, Varadhan, Xu & Yu 2000). The experiments using laptops running the implementation described in this thesis show that a range of 100 milliseconds results in much higher delivery latencies than those observed using the ns-2 simulator. Such a result is not surprising given the contemporary controversy surrounding the use of network simulators for evaluating wireless communication protocols. This is debated by Cahen et al. (2002), Andel & Yasinsac (2006) and Kurkowski et al. (2005), who show that results from different networks simulators diverge significantly

Tables 6.2 and 6.3 show the delivery latencies and the number of collisions observed for the different values of heartbeat used. The number of collisions is a rough estimate gathered from data made available by the *iwconfig* interface to the wireless card. This is a rough estimate because of the lack of documentation for the Cisco Aironet driver explaining the exact interpretation of the numbers available from the log file. The numbers shown in Table 6.3 is the sum of "excessive retries" and "miscellaneous invalid receptions", as per the documentation available. Excessive retries implies that a node backs off from transmitting a message more than the maximum allowable times (16 by default for the Linux Cisco

| Heartbeat(ms) | Delivery Latency (sec) | | |
|---|---|---|---|
| | *2 Nodes* | *4 Nodes* | *8 Nodes* |
| 100 | 0.02 | - | - |
| 500 | 1.99 | 2.00 | 4.62 |
| 1000 | 1.70 | 4.52 | 4.89 |
| 2000 | 1.48 | 2.01 | 3.68 |
| 4000 | 0.26 | 0.56 | 0.61 |

Table 6.2: Delivery Latencies

| Heartbeat(ms) | Number of Nacks | | |
|---|---|---|---|
| | *2 Nodes* | *4 Nodes* | *8 Nodes* |
| 100 | 5.88 | - | - |
| 500 | 2.47 | 9.92 | 23.62 |
| 1000 | 1.14 | 9.58 | 37.54 |
| 2000 | 0.63 | 2.13 | 24.43 |
| 4000 | 0.22 | 0.50 | 1.77 |

Table 6.3: Number of Negative Acknowledgements per node per sec

Aironet driver). These back offs are a result of high contention for the broadcast medium. Miscellaneous invalid receptions are interpreted as collisions at the receiving node, although these can be messages from a different MANET using the same broadcast channel. The sum of both these observations gives a rough estimate on the relative increase in contention for the broadcast medium.

The entries with a dash in Tables 6.2 and 6.3 imply that nodes in that experiment did not deliver messages from other nodes in the network. This reflects the high amount of congestion in the network for heartbeats of around 100 millisecond. This observation is in contrast with the parameters used for a study conducted on the ns-2 simulator, and highlights the importance of running experimental studies.

Experiments using only the broadcast protocol show that the heartbeat parameter used should be adapted with the size of the network. This reduces the delivery latency and also reduces the contention for the medium. For the purposes of the rest of the experiments the heartbeat is chosen as 500 $ms$ times the number of nodes in the network. An extensive theoretical analysis is required to determine the precise relationship between the network size, node density and system parameters like heartbeat. The approach taken here is a simple heuristic and provides initial estimates for the heartbeat values to use in the experiments.

### 6.3.2 Waitlength

Waitlength is used by the failure detector to suspect nodes as failed and is defined as the number of messages a node receives from each of the participating nodes, except those suspected. The failure detector can suspect a node in two ways, "failure to receive" and "failure to deliver" (as shown in Section 4.4). The waitlength used by the two methods can be different, but the experiments use the same waitlength, which is set equal to the square of the number of nodes in a group. Each node independently determines the waitlength, using the group view maintained at that group.

### 6.3.3 Duplicate Counter

Counter is the system parameter used by the "counter based optimisation" employed by the TransMAN implementation (Section 5.1.2). Tseng et al. (2002) proposed the use of counter based optimisation to reduce redundant broadcasts in a MANET. They propose the counter value be set to three for optimal results in a variety of network setups. However, because the experimental set up in the lab does not include networks with multiple hops the counter is set to one. That is, if a message is received more than once while it is awaiting transmission, it is not forwarded by the receiving node. Such an aggressive optimisation is used because the network setup in the lab has a high density of nodes, with up to eight nodes in the network.

## 6.4 Results

With the experimental setups described above, latencies for message delivery and stability are measured from two perspectives. First, for runs only with the broadcast protocol, and second for runs with the membership protocol. The goal is to measure the overhead of running the membership protocol on top of the reliable broadcast protocol.

The class *StaticMembership* is implemented (also shown in the class diagram in Figure 5.3 on page 97) to run the cases where only the broadcast protocol is executed. It is initialised with a known group membership that does not change during the system run. Dummy implementations are provided for various instance methods of the class that are called from the other classes in the system.

### 6.4.1 Observed Characteristics

A number of different performance characteristics are observed to present the performance of TransMAN protocols. This section lists the parameters and presents the motivation to include each of them in this evaluation study.

### Latencies

Delivery latency is measured as the average time elapsed between the time when a message is *sent* by a node and the different times when it is delivered at all participating node. Similarly the stability latency is the average time elapsed between the time when a message is *sent* by a node and when it stabilised at all nodes in the group. Comparing the delivery latency and stability latencies between the cases when the group membership protocol is running and when it is not running shows the delay overhead generated by the group membership protocol. It is expected that the use of TransMAN group membership protocol will not result in increasing the message latency such that the use of membership protocol renders the group communication useless. The usability of the group communication depends on the application being used and its latency tolerance levels. For the purpose of this evaluation less that 100% increase in delivery latency is deemed acceptable.

### Traffic Generated

Message latencies affect the applications and the end user of the system, and these latencies are in turn affected by the network traffic generated by the protocols providing the reliability and group communication guarantees. The generation of any extra traffic also adversely affects the battery lives of the mobile nodes running the protocol. Therefore observing the network traffic and how it is affected by the group membership protocol is important for capturing the utility of TransMAN.

Traffic generated while running the group membership protocol is compared to the traffic generated while the group membership protocol is not running, and is measured in terms of number of message transmissions in the network per second per node. Observing the traffic generated per node allows comparison of the traffic across networks of different sizes. The implementation of TransMAN is designed to minimise the extra traffic generated, and the results are expected to reflect this.

### Congestion Effects

Since the number of messages transmitted in a network is observed it is interesting to observe the congestion resulting from the traffic generated and measure any increase in congestion because of the membership protocol. An increase in network congestion will result in higher number of lost messages, further increasing the traffic in the network (because of negative acknowledgements generated) and may lead to node starvation.

Since IEEE 802.11b medium access control does not support collision detection, the congestion in the network is measured in terms of the number of negative acknowledgements sent by participating nodes. It is assumed that a higher number of negative acknowledgements implies a higher number of lost messages and thus a higher number of collisions in the network. It is expected that a higher node density of the network will result in higher congestion.

The number of cancellations of messages transmissions is also observed. A message cancellation implies a duplicate is received while it is queued for a retransmission. Cancelling the retransmission (as proposed by Tseng et al. (2002)) reduces the number of redundant transmissions of messages. Each message cancellation reflects the broadcast optimisation's attempt to reduce congestion in the network. It is expected that as the node density increases the number of message cancellations will increase. With a higher node density, a higher number of nodes forward a message, increasing the chances of receiving a duplicate and therefore cancelling the transmission. This attempt to reduce congestion is interpreted as an indicator for presence of network congestion, the higher the number of cancellations, the higher the expected congestion.

### Membership Reactivity

All the above characteristics are related to the communication overheads resulting from the use of the group membership protocol. Apart from limiting this communication overhead, the response times for group view changes should not increase rapidly with an increase in the group size. This is because the message stability protocol stops stabilising messages during a group view change, as discussed in Section 4.5.3, and the longer the delay in

installing a new view the larger the buffer sizes required to store delivered messages awaiting stabilisation.

Membership reactivity means the time taken by members of a group to install a new view if a node is added to or removed from the group. Membership reactivity depends on the sending rate of messages at all the nodes in the network, i.e. heartbeat. This is because membership agreement requires messages to be delivered from all the nodes confirming the agreement (see Section 4.5.1). The time for adding or removing nodes from a group will also depend on the size of the group. This is because the heartbeat adapts depending on the number of nodes in a group, i.e., as the network size increases the number of messages sent by a node decreases and thus the time taken to reach agreement will increase.

### 6.4.2 Observations

Each experiment is conducted 20 times and the results present the averages and the confidence interval for the measurements. Experiments are run for different group sizes, the heartbeat and waitlength are varied depending on the size of the group, and the counter value is kept constant. To generate transient changes to network connectivity the transmission ranges of half the nodes are varied every random interval between 10 to 20 seconds to a random value from the set of allowable values (5, 20, 30, 50mW).

Experiments for the group membership protocol are set up such that two nodes initially run for 180 seconds, then a third node joins. A group of three nodes is run for 180 seconds and then another node joins, and so on till all eight nodes have joined. This group of eight nodes is maintained for 180 seconds and then one node stops, reducing the group size to seven. A group of seven nodes run for 180 seconds and then another node stops, this continues till all nodes have terminated.

Latencies are observed for groups of various sizes during a group run for 180 seconds, both while nodes are being added to and removed from the network. Times to add and remove a node to a group of a certain size are measured from the same experiment. This setup is repeated 20 times, resulting in 40 samples of 180 seconds for each group size, 20 while nodes are being added and 20 while nodes are being removed from a group. The

experiments are run over night in an attempt to isolate affects of users using the wireless network in the department. Every night only five runs are executed, to reduce the affect of any disturbances from other networks in the lab. Repeating the same experimental set up and using randomised heartbeat delays, it is expected that results will capture the performance characteristics without being affected by external factors like network interference and topology.

### Latencies

It is expected that the membership protocol will not generate too high a overhead on top of the reliable broadcast protocol. This is because once tentative views are exchanged, the agreement protocol does not generate any extraneous messages, instead, the timeout messages from the reliable broadcast protocol (or application messages, if any) are used to reach agreement on group views.

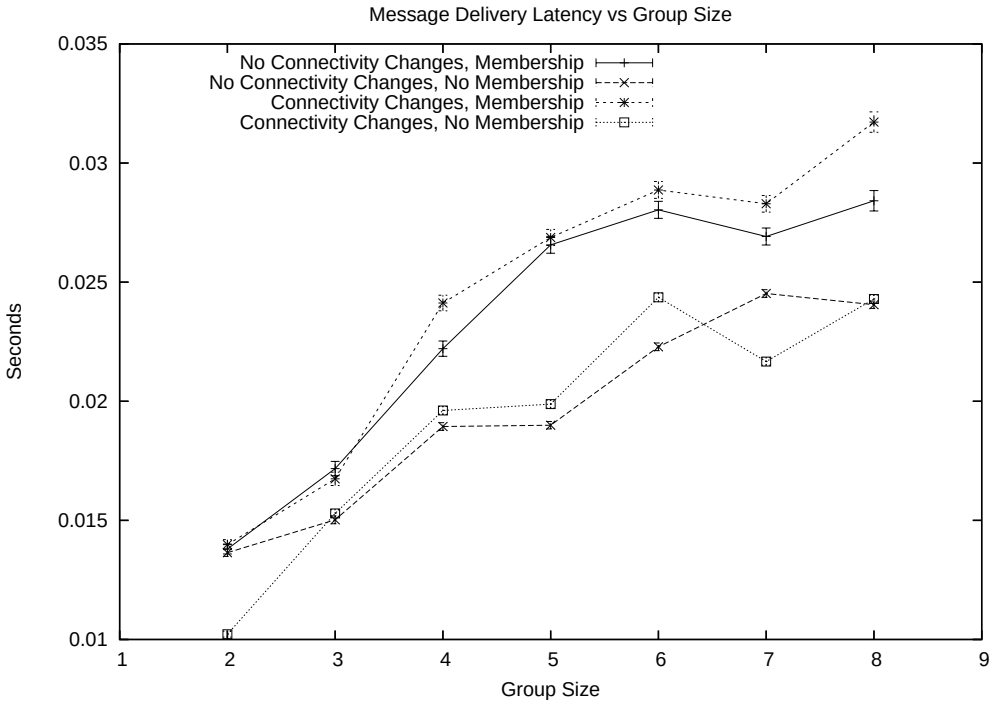


Figure 6.4: Delivery Latencies

Figure 6.4 shows the results of running the experiment without the membership service

and with the membership service for a network with and without changes in connectivity. The graph shows the group size on the x-axis and the message delivery latency (in seconds) on the y-axis. It is observed from the graph that delivery latency increases with increases in group size. This sub-linear relation is consistently observed for all variations in transient changes to network connections and use of the membership protocol. The delivery latency graph also shows that the use of membership protocol increases the delivery latency over the broadcast only by about 10 milliseconds, which is comparable to delivery latencies for group communication systems in LANs, WANs and MANETs (Mishra et al. 2002, Ezhilchelvan et al. 1995, Luo et al. 2004). A comparison of delivery latencies for networks with and without changes to network connectivity shows a very small absolute increase in latencies for changes in network connectivity, less that five milliseconds. This illustrates the broadcast protocol's ability to handle frequent changes in network connectivity.

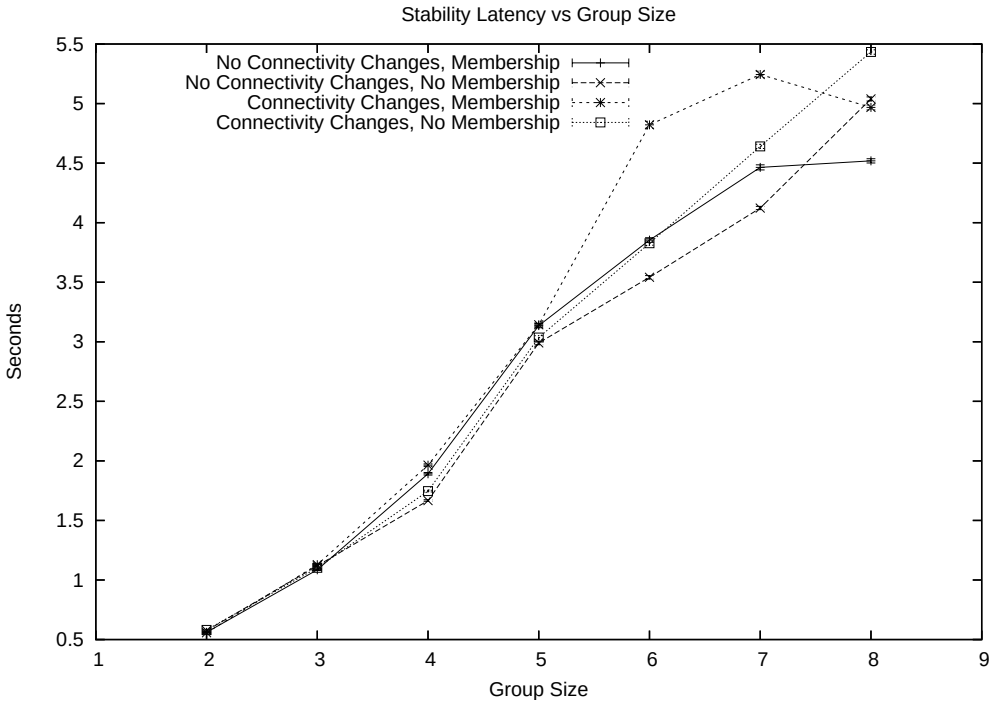


Figure 6.5: Stability Latencies

Figure 6.5 has the same parameters on the axis as Figure 6.4 but the times on y-axis show the times to stabilise messages using the TransMAN protocols. The message

stability latency is clearly dependent on the number of nodes in the group, increasing with an increase in group size. This is because the protocol gathers confirmations from all nodes to determine a message as stabilised. Since messages are stabilised in a total order (Section 4.3.3) the high absolute values (about 5.5 seconds) are a tradeoff for the total order provided. Some applications, for example distributed configuration management for a wireless networks (Barron, Weber, Clarke & Cahill 2005), require totally ordered updates and can tolerate high delivery latencies. Others can use the FIFO order message delivery which has latencies of tens of milliseconds (Figure 6.4).

The graph shows that the stability latencies of groups up to five nodes show minimal variations across network conditions and the use of membership protocol. However at group size of seven and eight nodes the stability latencies stop increasing for scenarios with the membership protocol while without the membership protocol the latencies increase with increase in group size. This is best explained if the traffic generated is observed as well: Figure 6.7 shows that the number of messages delivered per node per second increases with group size while using the membership service. This increase in message delivery rate results in a reduction in stability latencies. This is because as more messages are delivered each node is able to gather more implicit acknowledgements, providing faster stability determination of messages.

#### Traffic Generated

Figure 6.6 shows the number of messages transmitted per node per second for varying group sizes and the presence or absence of transient changes in network connectivity. The curves show the effect of adapting the heartbeat parameter depending on the size of the group. As the heartbeat is lowered the number of messages transmitted per node per second reduces. The Figure also shows that as the group size increases above three, the amount of traffic overhead resulting from the use of membership protocol increases. This increase is less than a 50% of the traffic generated by the broadcast protocol, and in absolute terms is less than one message per second per node. This small increase in traffic vindicates the use of the broadcast protocol for providing membership service.

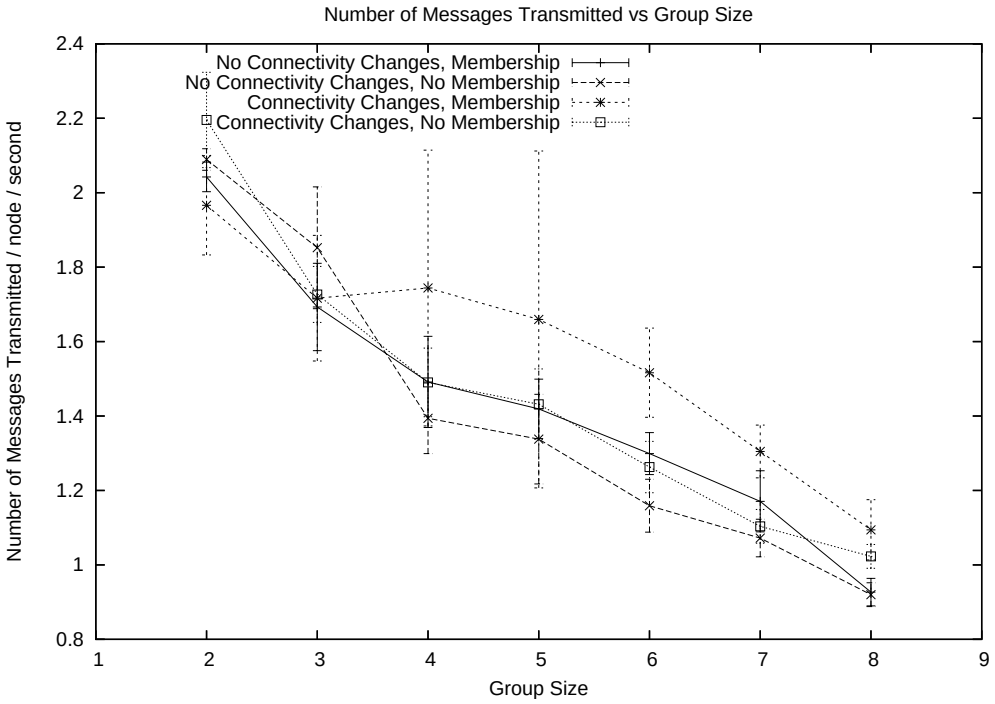


Figure 6.6: Number of Messages Transmitted per Node per Second

Figure 6.7 shows the number of messages delivered at each node per second. The case where membership is not used shows that the number of messages delivered per node per second remains nearly constant as the size of the network increases. From Figure 6.6 it is clear that the amount of traffic generated by each node decreases as the group size increases, and Figure 6.7 shows the reduction in traffic generation keeps the number of messages delivered constant. This is because of the reduction in traffic generated caused by adapting the heartbeat depending on the number of nodes in the network.

The important result from Figure 6.7 is that the number of messages delivered per node per second increases when the membership protocol is used. This is because of the increase in traffic generated by the use of the membership protocol (Figure 6.6).

The curve from Figure 6.7 for the case using the membership protocol also shows that the number of messages delivered per node per second increases with an increase in the number of nodes in the network. This is because even if the number of messages transmitted per node per second reduces, the increase in network traffic because of the

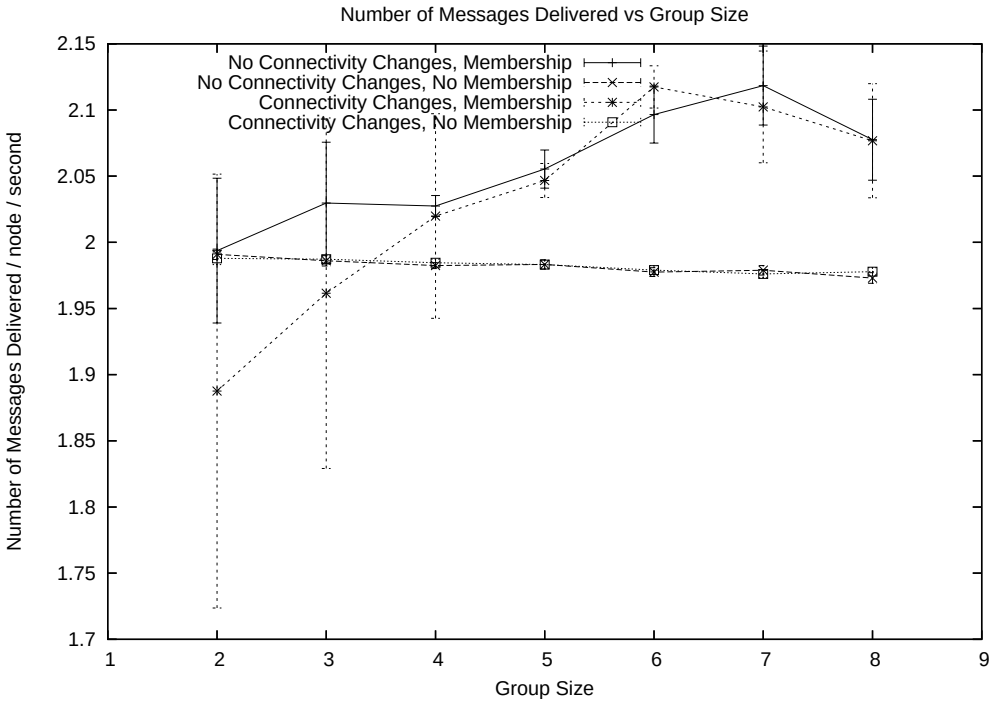


Figure 6.7: Number of Messages Delivered per Node per Second

membership protocol results in a higher number of messages delivered. As the number of messages in the network increases the participating nodes are able to detect lost messages sooner because some future message arrives that has a dependency on the lost message (see Section 4.2.2). This observation shows that a higher traffic in the network increases the rate of message delivery at nodes.

Figure 6.8 shows that the number of messages stabilised per node per second as compared to the group size and presence or absence of transient changes to network connectivity. The graph shows that number of messages stabilised per node per second remains constant when the group membership protocol is not run. This is explained by the slower heartbeat used for a larger network that reduces the number of messages transmitted when the group membership protocol is not running (Figure 6.6). However when the group membership protocol is running the number of messages stabilised per node per second shows a variation around the values for the no membership case, with a trend of first increasing and then decreasing message stabilisation rate. This trend is similar to the one for message

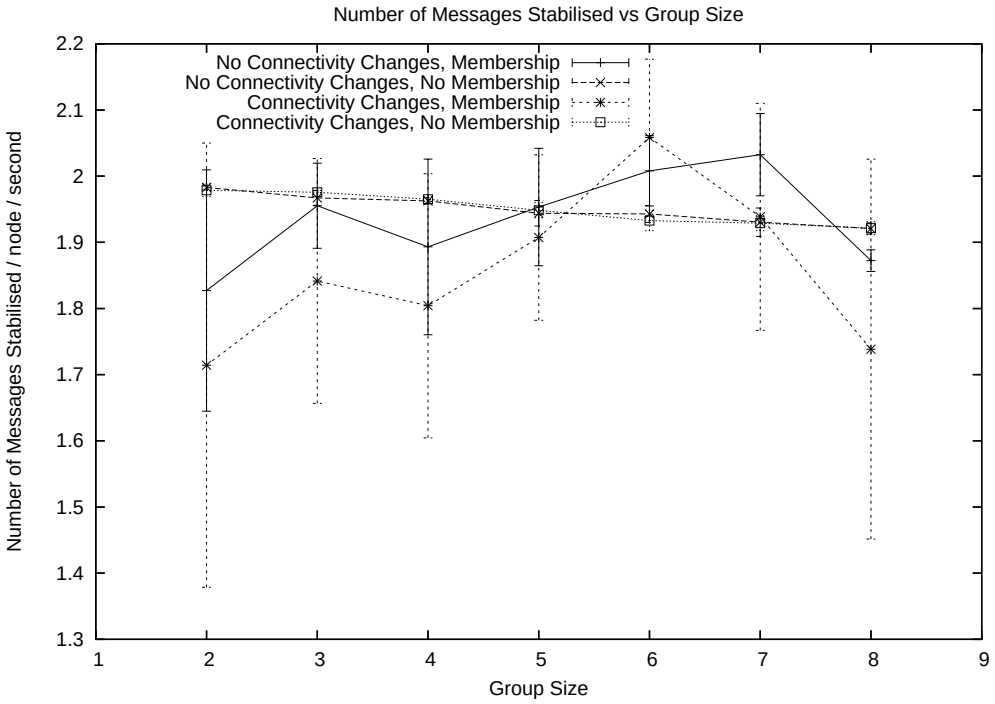


Figure 6.8: Number of Messages Stabilised per Node per Second

delivery rate, leading to the conclusion that as expected the messages stability is affected by the message delivery rate. This is expected because determining message stability requires that all nodes deliver messages from all other nodes that allow determination of a message's delivery status. A broadcast optimisation that provides efficient delivery of messages will result in an improvement in TransMAN's message stability rate as well.

### Congestion Effects

Figures 6.9 and 6.10 show the number of negative acknowledgements and the number of transmissions cancelled per node per second for varying group sizes and presence or absence of transient changes to network connectivity. The figures show that the number of negative acknowledgements and cancellations for the membership and no membership are statistically equivalent because of the overlapping confidence intervals for the curves. This shows that the membership protocol does not result in an increase in network congestion.

Figure 6.9 shows that the network with changes in connectivity has a higher mean

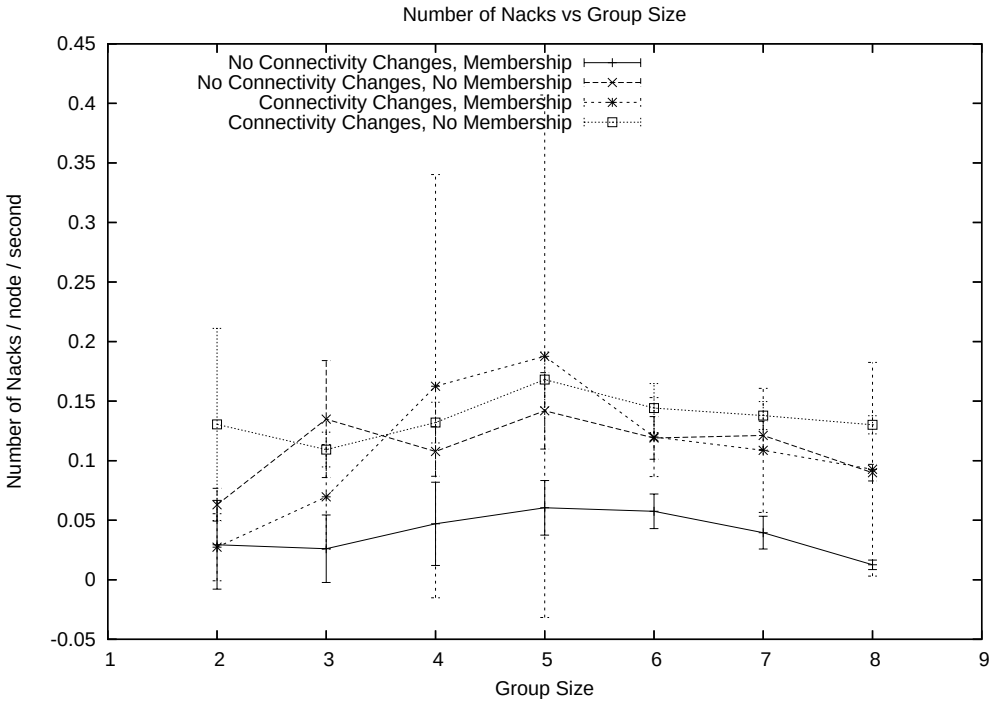


Figure 6.9: Number of Negative Acknowledgements per Node per Second

number of nacks. The case with the membership protocol and network changes shows a large variation in the number of nacks generated. These observations are as expected, a network with transient network conditions and handling a higher amount of traffic (from the membership protocol) results in a higher number of message losses. The broadcast protocol is, however, still able to handle the changes in network connectivity without a reduction in the rate of message delivery (Figure 6.7).

Figure 6.10 shows the number of message transmissions that are cancelled by the broadcast optimisation while they are awaiting retransmissions. The graph shows that as the node density increases the number of transmissions cancelled increases, as expected because of the increase in network traffic. The graph also shows that the number of cancellations are higher for the network with transient changes to network connectivity, which is caused by the higher traffic and higher number of nacks for these cases.

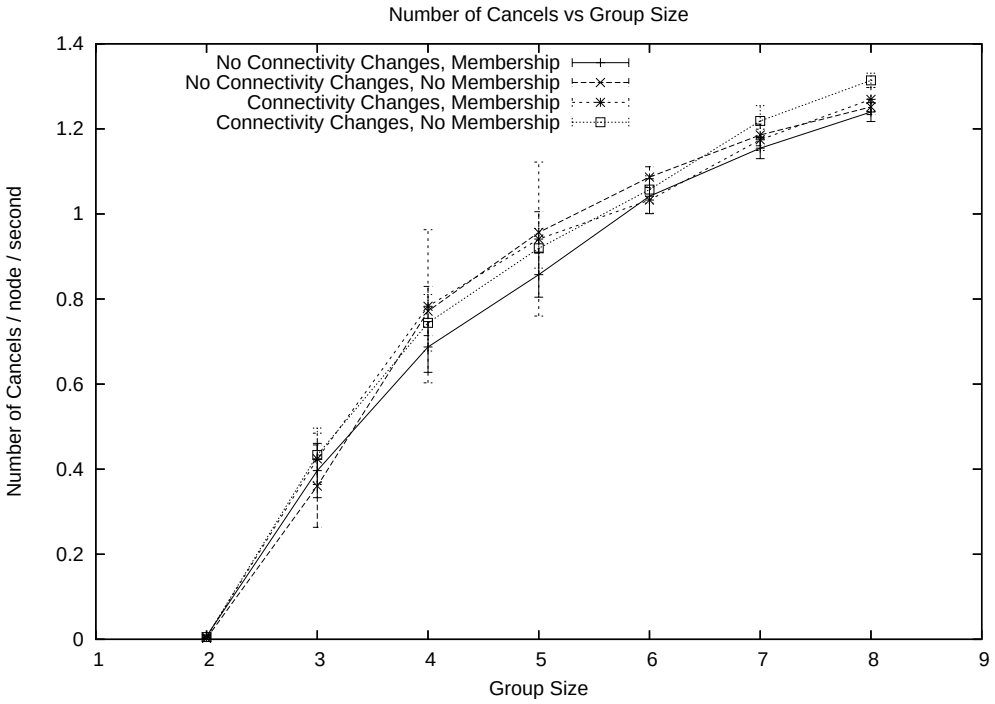


Figure 6.10: Number of Transmissions Cancelled per Node per Second

### Membership Reactivity

Figures 6.11 and 6.12 show the time required to add and remove one node from an existing group of different sizes. These times reflect the time to propose a new group view (phase I of the agreement protocol) plus the time to reach agreement (phase II of the agreement protocol) on a new group view. The presence and absence of transient changes to network conditions are considered in both graphs. The high absolute values for adding and removing a node, especially for large groups, is a result of the heartbeat adaptation to reduce network congestion. These high values encourage the use of better broadcast optimisations that can reduce congestion to allow smaller heartbeat values.

The graphs show that the time to reach agreement on group view change increases with an increase in group size. This is because agreement on a group view requires that all nodes send confirmations to the proposed group views and they all are able to determine that all other nodes have received all the confirmations. The time required to gather all

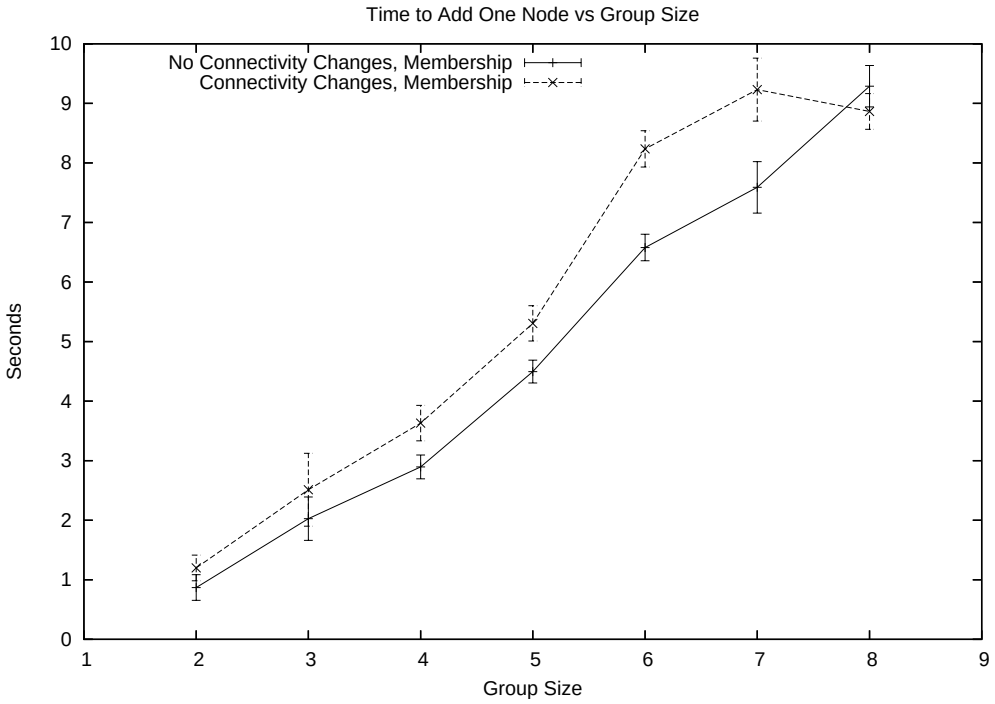


Figure 6.11: Time to Add a Node

these confirmations is dependent on the group sizes and is reflected in the increase in the time to change group views.

The small difference in the times to reach agreement when transient changes are present or not shows that TransMAN is able to handle the transient changes without resulting in high delays for group view changes.

Finally, when comparing the two times to add or remove a node for a given group size it is observed that the time to remove a node is larger than the time to add a node, and the difference increases as the size of the network increases. This is because of a difference in the first phase of the protocol when a node is added to or removed from a group. In the case when a node fails, all nodes are required to determine the failure independently. When a node is being added to a group, the first node detecting a new node triggers all nodes to propose the next group views (see Section 4.5.1).

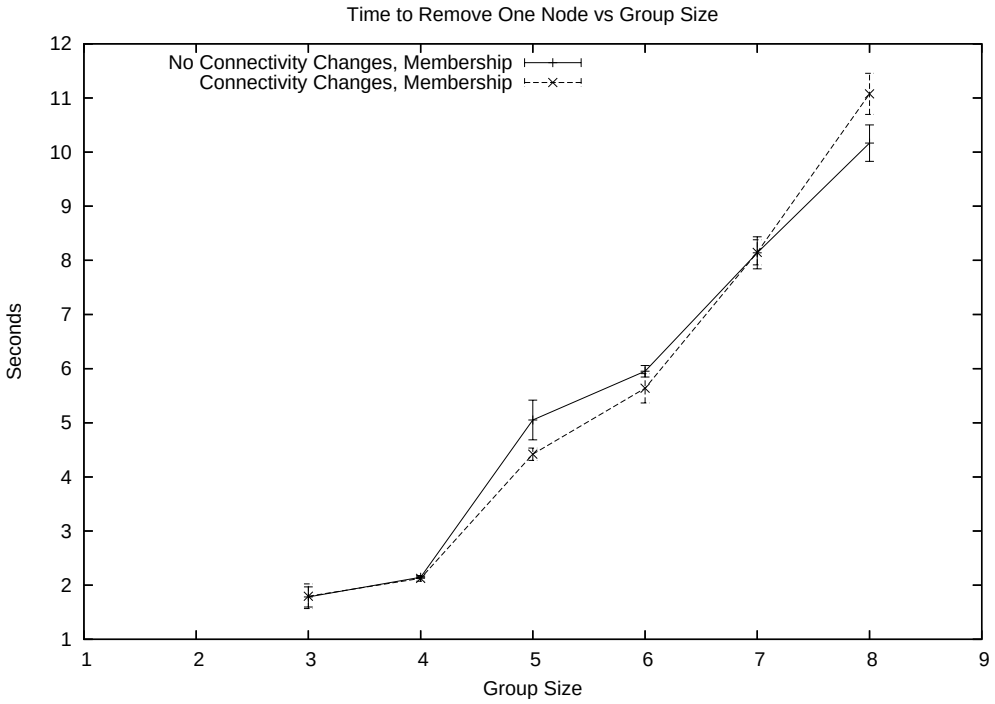


Figure 6.12: Time to Remove a Node

## 6.5 Conclusions

This chapter described the experimental setup and the various system parameters used to evaluate an implementation of the TransMAN group communication system. The aim of the study was to observe the overhead of the group membership protocol over the reliable broadcast protocol used by the system. The functional properties supported by the TransMAN group communication protocols (i.e., reliable broadcast, ordered delivery of messages, consistent list of participating nodes, non-blocking view changes and virtual synchrony) were proved in Section 4.6. This chapter showed an evaluation study that measured the overhead of providing all these functional properties as compared to providing only reliable broadcast and ordered delivery of messages.

Initial experiments to decide on the heartbeat values to use in networks of different sizes showed that the heartbeat values used in an earlier study using the ns-2 simulator were highly unsuitable for use in this study. The heartbeat values used in the simulator study

results in higher congestion leading to messages not being delivered at all participating nodes. This showed that experiments from simulator studies and real world studies provide very different results.

The overhead of the membership protocol is measured in terms of added delays to message delivery and stabilisation latencies, increase in amount of traffic generated, and increase in the network congestion by running the membership protocol. The conclusions from the performance study are summarised in the Table 6.4. The salient lessons from these conclusions are that (1) total order delivery comes at the cost of increased message latency, (2) total order delivery, message stability and membership agreement are influenced by the rate of message delivery, (3) the rate of message delivery is affected by the amount of traffic generated by each of the participating nodes, and (4) the FIFO delivery latency is in tens of milliseconds (but at the cost of a reduced message generation rate).

The affect of the rate of message delivery on message stability and membership agreement means that the choice of broadcast optimisation becomes important in the performance of higher layers of TransMAN. The simple broadcast optimisation used in the study presented in this thesis is able to handle only a low heartbeat of message generation, especially for large networks. This results in high stability latencies and increased membership reactivity for large networks.

However, the overhead resulting from the use of the membership protocol is always less than a 100% increase. This the most important conclusion from the performance results and it vindicates the use of application broadcast messages to provide membership agreement, even at the expense of a higher computation overhead.

| **Characteristic** | **Conclusions** |
|---|---|
| Delivery Latencies | 1. The use of the membership service increases message delivery latency, as expected, but the overhead is minimal of the order of 10 milliseconds.<br>2. Transient changes in network connectivity results in only a marginal increase in delivery latency. |
| Stability Latencies | 1. Transient changes in network connectivity result in higher stability latencies.<br>2. A higher rate of message delivery results in a reduction in stability latencies. |
| Traffic Generated | 1. The membership protocol results in less than twice the amount of traffic generated when compared to broadcast only cases.<br>2. A higher amount of traffic in the network results in a higher rate of message deliveries. |
| Congestion Effects | 1. Transient changes to network connectivity result in a higher number of message losses and also a higher number of message transmissions cancelled.<br>2. A higher loss rates is accompanied by a higher rate of transmission cancellations.<br>3. The use of membership protocol does not increase the congestion in the network. |
| Membership Reactivity | 1. Time to reach agreement on a group view is proportional to the group view size.<br>2. Time to remove a node from a group is higher than the time to add a node to a group of a same size. |

Table 6.4: Conclusions Made From The Observations

# Chapter 7

# Conclusions

This chapter summarises the contributions made by this thesis and makes suggestions towards possible improvements to and uses of TransMAN.

## 7.1 Contributions

This thesis presented a design, implementation and an evaluation study for a group communication in MANETs, called TransMAN. The thesis recognised the challenges posed by a dynamic environment like MANETs and the design of TransMAN was motivated to address these challenges.

The use of existing solutions from LANs and WANs is unattractive because MANETs have a different network model. In MANETs, mobile nodes provide broadcast domains and have the ability to send a message to a number of nodes with a single transmission. Unlike those in a LAN, these transmissions are highly susceptible to failure because MANET medium access control protocols do not provide a collision detection facility that can be used for a reliable broadcast message transmission. In addition, node mobility and limited battery lives affect network connectivity by creating transient changes.

TransMAN utilises optimised broadcasts and implicit acknowledgements to reduce the number of message transmissions in a MANET. This reduces the battery consumption at mobile nodes, but required a tradeoff that increased the processing overhead of handling

each message reception. The increased processing is a result of the use of a message dependencies graph that enables the provision of implicit acknowledgements. These are used to provide reliable delivery, message stability and membership agreement.

The use of a graph of message dependencies was motivated by the idea of utilising the "broadcast domain" provided by wireless nodes and using a broadcast protocol to allow a number of overlapping broadcast domains to act as a single broadcast domain. This allows message delivery and group membership properties to be provided across all nodes present in the overlapping broadcast domain, and is achieved by using the graph of message dependencies to track implicit acknowledgements across the composite broadcast domain.

Message delivery is supported in FIFO and total orders and applications have a choice on what ordering they want to use. Total order delivery of messages came at a cost of high delivery latency because messages are delivered in total order only when nodes agree on the next message in the total order. This agreement protocol requires that messages be delivered from all nodes, substantially increasing the message delivery latency when compared to FIFO order delivery. TransMAN also provides a message stability protocol and messages are stabilised in total order.

A failure detector is provided to facilitate initialisation of a group view change protocol when a member node fails. This failure detector runs on all nodes and allows nodes to independently suspect a node as failed. Suspicions of node failures result in initiation of a membership agreement protocol where all nodes propose a group view, reaching an agreement on a view that they all install.

The membership agreement protocol allows each node to keep proposing views and manages a number of such agreements on all the nodes in parallel. This is facilitated by the use of application and timeout messages to reach agreement and the use of a message dependency graph to determine agreement termination. The membership agreement protocol tracks agreement on a number of group views without generating additional traffic. This is achieved by the use of message dependencies graph and implicit acknowledgements to track agreement termination.

The design presented handles transient changes in network connectivity without block-

ing applications during periods of transient changes. For example, if a group of mobile devices are participating in a collaborative work application and a node that is not in the group temporarily enters the radio ranges of any of the nodes in the group, the collaborative application is not put on hold (or blocked) while the new node either joins the group or leaves the radio range of the nodes in the group.

If the number of nodes transiently connecting and disconnecting from the MANET is high, the nodes initiate agreement on all the transient group views that they can propose. This approach makes TransMAN a best effort service, if and when a transient view is stable enough that an agreement is reached on it, the view is installed as the new view. While transient view changes initiate agreement on proposed group views, the members that are in a group and do not disconnect from each other continue in the group communication, without being interrupted by the new membership agreements.

The message delivery (FIFO and total) and the membership agreement protocols in TransMAN conform to a list of group communication properties (shown in Appendix A) carefully chosen from existing frameworks (Babaoğlu et al. 2001, Chockler et al. 2001). These guarantee that the group communication service provides well understood properties. By supporting these properties, TransMAN supports a consistent list of participating nodes at all the nodes and a virtually synchronous group communication system where all nodes that survive the same group view changes deliver the same set of messages.

An evaluation study of the TransMAN group communication service showed that the the overhead from the use of the membership protocol is always less than a 100% increase in traffic generated or message delivery latencies over those from only running the broadcast protocol. The experimental study also showed that (1) total order delivery comes at the cost of increased message latency, (2) total order delivery, message stability and membership agreement are influenced by the rate of message delivery, (3) the rate of message delivery is affected by the amount of traffic generated by each of the participating nodes, and (4) the FIFO delivery latency is in tens of milliseconds (but at the cost of a reduced message generation rate). The affect of the rate of message delivery on message stability and membership agreement means that the choice of broadcast optimisation becomes im-

portant in the performance of higher layers of TransMAN. However, the overhead resulting from the use of the membership protocol is always less than a 100% increase to the amount of traffic or latencies without the use of membership service.

## 7.2 Future Work

While implementing and evaluating TransMAN a number of possibilities for its use and its improvements were identified, as follows:

### Using Improved Broadcast Optimisation

The evaluation study was carried out using a very simplistic broadcast optimisation protocol, i.e., the counter based optimisation (Tseng et al. 2002). An evaluation study with a broadcast optimisation that utilises network information to provide better optimisation will result in more detailed insights into the overheads of the GCS protocol.

### Fair Broadcasts

The *eventual fair broadcast* assumption made while implementing TransMAN is not supported by the IEEE 802.11b medium access protocol. Unfairness of the broadcast protocol results in an imbalance in the messages delivered from different nodes. This was reflected in the evaluation study where an increase in traffic did not generate a relative increase in number of messages stabilised. Using the group membership available and some additional information like location of the nodes, a broadcast protocol can be implemented that can avoid the unfairness problem in MANETs (la Kloul & Valois 2005, Barrett et al. 2002, Bensaou, Wang & Ko 2000, Singh 2006).

### Broadcast Optimisation That Uses TransMAN

Existing broadcast optimisations try and gather information from and about the MANET, using services like location dissemination to inform the broadcast optimiser so it can take better and more informed decisions about cancelling a message transmission. The use of

group membership information and also the information about transient views at a node can be used to provide a broadcast optimisation that will outperform the existing solutions.

### Fuzzy Message Stability

The TransMAN group communication tracks a number of transient views at the same time to handle transient changes in network connectivity. This information can be used to support the idea of providing fuzzy message stability, as proposed by Friedman & Tcharny (2003a). This will provide quicker message stabilisations, reducing the amount of buffer space required at each node.

### Congestion Control

Similar to the proposal to use TransMAN's membership and transitive view information to provide broadcast optimisations, a scheme for congestion control in MANETs is possible. This solution would allow nodes in a MANET to change the amount of traffic they generate based on the view of the network made available by providing membership and transient views (Singh 2006). How such a congestion control scheme can be achieved in a MANET with transient changes to network connectivity is a challenge (Tang, Obraczka, Lee & Gerla 2002).

### Theoretical Analysis

Theoretical analysis of a number of broadcast protocols for MANETs has been undertaken by making assumptions that do not reflect a real world MANET. However, such analyses are useful for relative comparisons of these broadcast protocols. A theoretical analysis of the extra traffic generated by TransMAN's group communication service would help identify optimisations for the protocol, providing lower message stability and membership reactivity times.

### Transport Layer Group Communication?

If the complexity of the data structures presented can be implemented in device drivers, a group communication system for MANETs at the transport layer would result in new ways of using MANETs. These would allow for applications that can depend on the reliability of the MANET communication and can expect formation of groups to achieve collaborative tasks. A MANET transport layer group communication could provide a LAN-like abstraction, where if a node is turned, on it is immediately available in "abstract MANET LAN" where nodes in the abstract LAN have consistent lists of participating nodes, as long as nodes remain reachable from each other.

### State Transfer Protocol

When partitions of a group merge to form a larger group, applications using the group communication service need to exchange their state to synchronise the states across all nodes. A protocol to support state transfer can be crucial to support some applications, for example, applications that support collaborative work. Such a state transfer protocol is not yet provided with TransMAN and remains a problem that needs to be addressed.

### Experiments With Multi Hop MANETs

The evaluation study presented in this thesis is conducted in a lab environment where changes in network connectivity are effected by varying the transmission power of the nodes. Such a set up allowed repeatable experiments at the cost of limiting the MANET to a single hop. The aim of the study was to observe any overhead resulting from the use of TransMAN GCS protocols.

It can be argued that varying the transmission power of nodes must have resulted in some messages being forwarded over more than one hop. But the fraction of messages that are forwarded is not not easily deducible, unless a stochastic analysis addresses the issue. A study where nodes are either set up in a liner topology or are mobile so that the network topology changes and messages are sent over more than one hop would provide more insights into the overheads resulting from the use of TransMAN GCS.

# Appendix A

# System Specifications

This appendix lists the properties that specify the TransMAN group communication system presented in this thesis. The specifications provided by borrowing system properties from the frameworks presented by Chockler et al. and Babaoğlu et al.. Table A.1 lists short predicates that are useful while describing the properties of TransMAN.

### A.0.1 Membership Service

**Self Inclusion** – If a process $p$ installs view $V$, then $p$ is a member of $V$.

$$\exists i \; t_i = \; \mathbf{view\_chng}(p, V, T) \quad \Rightarrow \quad p \; \in \; V \tag{A.1}$$

The property eliminates situations where a node suspects itself or participates in a group whose members think the node has failed.

**Local Monotonicity** – If a process $p$ installs view $V$ after installing view $V'$ then the identifier of $V$ is greater than that of $V'$.

$$\begin{aligned} t_i = \mathbf{view\_chng}(p, V, T) \quad \wedge \quad & t_j = \mathbf{view\_chng}(p, V', T) \\ \wedge \, i > j \quad \Rightarrow \quad & V.id > V'.id \end{aligned} \tag{A.2}$$

The property makes sure that a process does not install the same view twice and if two processes install the same views they install them in the same partial order.

| | |
|---|---|
| $delivers(q, m)$ | $\exists i\, t_i = \mathbf{recv}(q, m)$ |
| $delivers\_in(q, m, V)$ | $\exists i\, t_i = \mathbf{recv}(q, m)\ \wedge\ viewof(t_i) = V$ |
| $installs(p, V)$ | $\exists i\, t_i = \mathbf{view\_chng}(p, V, T)$ |
| $viewof(t_i)$ | $existsj < i\ s.t.\ t_j = \mathbf{view\_change}(p, V, T)$ |
| $installs\_in(p, V, V')$ | $\exists i\, t_i = \mathbf{view\_chng}(p, V, T)\ \wedge\ viewof(t_i) = V'$ |
| $recv\_before(p, m, m')$ | $\exists i \exists j\ (t_i = \mathbf{recv}(p, m)\ \wedge\ t_j = \mathbf{recv}(p, m')\ \wedge\ i < j$ |
| $recv\_before\_in(p, m, m', V)$ | $\exists i \exists j (t_i = \mathbf{recv}(p, m) \wedge t_j = \mathbf{recv}(p, m') \wedge viewof(t_i) = viewof(t_j) = V\ \wedge\ i < j$ |

Table A.1: TransMAN Specifications: Often Used Short Predicates

**Initial View Event** – Every send or recv event occurs within some view.

$$t_i = \mathbf{send}(p, m) \vee t_i = \mathbf{recv}(p, m) \vee \quad \Rightarrow \quad viewof(t_i) \neq \perp \tag{A.3}$$

The property requires that all messages are exchanged in the context of some group view and places the requirement that the first message sent by a node is sent in a view consisting of the node itself.

**Membership Accuracy** – If there is a time after which processes $p$ and $q$ are alive and are reachable from each other, then $p$ eventually installs a view that includes $q$, and every view that $p$ installs afterwards also includes $q$.

$$\begin{aligned}
\exists i \text{ s.t } \wedge\ \forall k < i \quad & t_k = \mathbf{reachable}(p, q) \\
\wedge \forall\, l > i \quad & t_l \neq \mathbf{crash}(p) \\
\wedge \forall m > i \quad & t_m \neq \mathbf{unreachable}(p, q) \\
\Rightarrow \exists j\ \exists V \text{ s.t.} \quad & t_j = \mathbf{view_chng}(p, V, T) \wedge q \in V.members \\
\wedge \forall k > j\ \forall V' \quad & t_k = \mathbf{view_chng}(p, V', T)\ \wedge\ q \in V'.members
\end{aligned} \tag{A.4}$$

This property eliminates solutions to the group membership where nodes $p$ and $q$ are

reachable from each other (directly or indirectly) and neither of them installs a view with both as members of the view.

Membership Accuracy does not require that nodes that are connected to each other should install the same view, it only requires these nodes to include each other in their view. Installing the same view requires another property presented later as *View Coherency* with equations (A.6) and (A.7).

**Termination of Delivery** – [1] If a process $p$ sends a message $m$ in a view $V$, then for each member $q$ of $V$, either $q$ delivers $m$, or $p$ installs a next view $V'$ in $V$.

$$\begin{aligned} \exists\, l \text{ s.t.} t_l = \mathbf{send}(p,m) \quad &\wedge \quad viewof(t_l) = V \\ &\wedge \quad q \in members(V) \\ \Rightarrow delivers(q,m) \quad &\vee \quad \exists\, V'\, installs\_in(p,V',V) \end{aligned} \tag{A.5}$$

This property captures the liveness of the membership service by requiring that if a message is not delivered at some member of a view (implying that the other member is no longer reachable), then the sender of the message installs a new view with the failed process removed from the view. Thus, the group communication service is forced to make progress.

**View Coherency** – The property can be presented in two steps.

1. If a correct process $p$ installs a view $V$, then either all members of $V$ install $V$ or $p$ eventually installs an immediate successor to $V$.

$$\begin{aligned} &\exists i\ t_i = installs(p,V) \\ \Rightarrow\ \forall\, q \in V.members\ installs(q,V) \quad &\vee \quad t_{i+1} = installs(p,V') \end{aligned} \tag{A.6}$$

2. If two nodes $p$ and $q$ initially install the same view $V$ and $p$ later installs an immediate successor to $V$, then eventually either $q$ also installs an immediate

[1] This property also implies a *Self Delivery* property, which states that a correct process delivers messages sent by itself.

successor to $V$, or $q$ crashes.

$$\begin{aligned} & \exists i\, t_i = installs(p, V') && \wedge \quad \exists j < i\, t_j = installs(p, V) \\ & && \wedge \quad \exists k < i\, t_k = installs(q, V) \\ \Rightarrow\ & \exists l > i\, t_l = installs(q, V') && \vee \quad t_l = \mathbf{crash}(q) \end{aligned} \tag{A.7}$$

The situation handled by this property can be argued to have been handled by *Membership Accuracy*, but *Membership Accuracy* leaves the installation of the new view totally dependent on a failure detector. View Coherency forces all members of any new view install the same view.

A important aspect of view installation is the use of *Transitional Sets* to support *Virtual Synchrony*. These are discussed once the properties of message delivery service have been discussed because *virtual synchrony* specifies the behaviour of both the membership service and message delivery service during view changes.

### A.0.2 Delivery Service

This section specifies the behaviour of TransMAN about delivering messages. These specifications map to the reliable broadcast requirements from Section 1.1 (see also Table 3.2).

**Delivery Integrity** – For every **recv** event, there is a corresponding **send** event.

$$t_i = \mathbf{recv}(p, m) \quad \Rightarrow \quad \exists q\ \exists j \text{ s.t. } j < i\ \wedge t_j = \mathbf{send}(q, m) \tag{A.8}$$

**No Duplication** – Two different **recv** events with the same content cannot occur at the same process.

$$t_i = \mathbf{recv}(p, m)\ \wedge\ t_j = \mathbf{recv}(p, m) \quad \Rightarrow \quad i = j \tag{A.9}$$

The above two properties together eliminate spurious *recv* events. Limiting *recv* events to only those that originated at some node and already been delivered.

**Self Delivery** – Process $p$ delivers every message it sent in any view, unless it crashed after sending it.

$$t_i = \mathbf{send}(p, m)\ \wedge\ \nexists j > i\, t_j = \mathbf{crash}(p) \quad \Rightarrow \quad \exists\, k > i\, t_k = \mathbf{recv}(p, m) \tag{A.10}$$

**Virtual Synchrony**

The property of virtual synchrony specifies the behaviour of the membership and message delivery services during a group view change. The Section 2.3.3 on page 42 described the various models of virtual synchrony. The requirement of virtual synchrony specified here along with the use of transitional sets maps to the *Extended Virtual Synchrony* model presented by Moser, Amir, Melliar-Smith & Agarwal (1994).

**Virtual Synchrony** – If process $p$ and $q$ install the same view $V$ in the same previous view $V'$, then any message received by $p$ in $V'$ is also received by $q$ in $V'$.

$$\begin{aligned} installs\_in(p,V,V') \wedge \quad installs\_in(q,V,V') \quad &\wedge delivers\_in(p,m,V') \\ \Rightarrow \qquad\qquad\qquad\quad delivers\_in(q,m,V') \quad &\text{(A.11)} \end{aligned}$$

Virtual synchrony is a useful guarantee but has the limitation that it is an external observer property. Nodes can not determine if the requirements of virtual synchrony are satisfied during a group view change. If group view change is not virtually synchronous, applications that require all nodes to be in a consistent state have to go through a state transfer protocol. To allow nodes to determine if virtual synchrony holds during a group view change, the nodes are provided with *Transitional Sets* between the view changes.

Transitional sets delivered at a node tell the node which other nodes have a state identical to itself. If there are nodes in the new view which were not in the transitional set delivered before the view change, the nodes need to undergo a state transfer to achieve identical states. The provision of transitional sets in TransMAN is discussed in Section 4.5.2.

**Transitional Sets** – 1. If process $p$ installs a view $V$ in (previous) view $V'$, then the transitional set of view $V$ at process $p$ is a subset of the intersection between the member sets of $V$ and $V'$.

$$\begin{aligned} t_i = \mathbf{view\_chng}(p,V,T) \ \wedge \ viewof(t_i,p) = V' \\ \Rightarrow T\ V.members \ \cap \ V'.members \qquad \text{(A.12)} \end{aligned}$$

2. 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.

$$t_i = \mathbf{view\_chng}(p, V, T) \quad \wedge viewof(t_i, p) = V' \quad \wedge installs\_in(q, V, V'') \\ \Rightarrow \ (q \in T \Leftrightarrow V' = V'') \tag{A.13}$$

**Message Ordering**

Once the membership service and message delivery satisfy the properties listed above a distributed application using TransMAN is guaranteed to deliver the same set of messages in the same group views. However the application can make use of the messages only if messages are delivered in a well known order. Unordered delivery of messages does not provide the application with any context for a received message and therefore the application is unable to utilise the messages. TransMAN supports both FIFO and Weak Total Order message delivery and this section describes both these ordering schemes.

These message ordering properties are defined by using the notion of a Timestamp (TS) function as defined in (Chockler et al. 2001). A timestamp (TS) function is a one-to-one function from $\mathcal{M}$ to the set of natural numbers:

$$TS\_function(f) \Longrightarrow f \ : \quad \mathcal{M} \rightarrow \mathcal{N} \ \wedge \ f(m) = f(m') \quad \Rightarrow m = m'$$

**Reliable FIFO** – If process $p$ sends messages $m$ before message $m'$ in the same view V, then any process $q$ that receives $m'$ receives $m$ as well.

$$t_i = \mathbf{send}(p, m) \quad \wedge t_j = \mathbf{send}(p, m') \quad \wedge i < j \ \wedge \ viewof(t_i, p) = viewof(t_j, p) \\ \wedge receives(q, m') \quad \Rightarrow \ receives(q, m) \tag{A.14}$$

**Weak Total Order** – Can be stated in two parts

1. 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

$f$.

$$\begin{aligned} \forall\, V\ \forall V'\ \exists f(TS\_function(f) \quad &\wedge \quad \forall p\ \forall m\ \forall m'\ (installs\_in(p, V, V') \\ &\wedge \quad recv\_before\_in(p, m, m', V') \\ &\Rightarrow \quad f(m) < f(m')) \end{aligned} \tag{A.15}$$

2. 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 –

$$\begin{aligned} \forall V\ \exists f\ (TS\_function(f) \quad &\wedge \quad \forall p \forall m\ \forall m'(last\_view(p, V) \\ &\wedge \quad recv\_before\_in(p, m, m', V) \\ &\Rightarrow \quad f(m) < f(m'))) \end{aligned} \tag{A.16}$$

*Weak Total Order* allows nodes to disagree on the order of messages in case they disconnect from each other. However, *Weak Total Order* requires nodes that remain connected with each other to receive messages in the same order.

# Appendix B

# Correctness Proofs For Broadcast Protocol

This appendix presents the proof for Theorem 5 on page 68. Consider four cases illustrated in Figure B.1. Cases $(i)$ and $(ii)$ assume static nodes and induction is used to prove the theorem under the static nodes assumption. Cases $(iii)$ and $(iv)$ allow for mobile nodes and again an induction is shown.

If a node $p$ sends a message $p_i$, nodes that stay in the immediate neighbourhood of $p$ eventually receive $p_i$. This is the case in Figure B.1 $(i)$.

**case (i)** From the eventual fair broadcast assumption, $q$ eventually receives some message $p_k$ from $p$, such that $k > i$. From Lemma 1, $p_i \triangleright p_k$. From rule $\mathcal{R}3$, $q$ nacks for some dependency of $p_k$. By the definition of dependencies, $q$ eventually nacks for $p_i$ and from the eventual fair broadcast assumption, receives it.

The assumption of static nodes is used to prove that if a node $p$ sends a message, all nodes that remain reachable from $p$ receive the message. The proof is an induction on the series of neighbouring nodes that a message goes through.

**case (ii)** Figure B.1 $(ii)$ presents this case. If the node $p$ sends a message $p_i$, from case $(i)$, $q$ eventually receives the message. This is the base case for the induction. As the inductive step, it is assumed that the $(k-1)^{th}$ neighbour receives the message

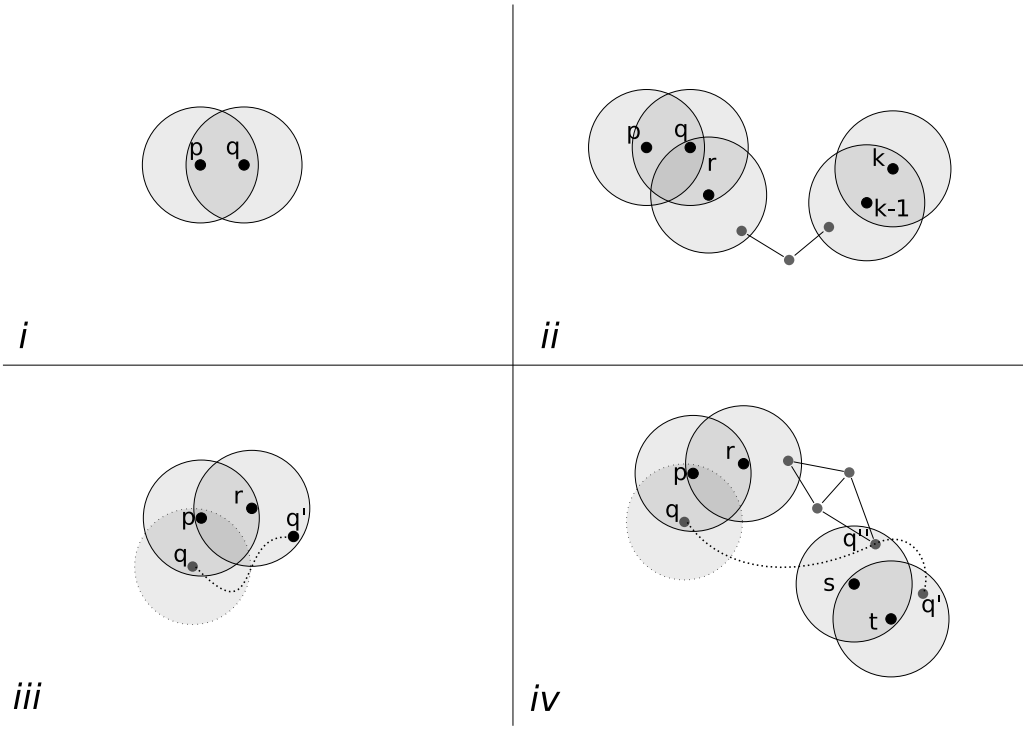


Figure B.1: Node $p$ sends a message, $q$ eventually receives the message.

and a proof is presented to show that whenever the $(k-1)^{th}$ neighbour receives $p_i$, the $(k)^{th}$ neighbour receives $p_i$. Assume that the $(k-1)^{th}$ and $k^{th}$ neighbours have identities $k-1$ and $k$, as shown in Figure B.1. From Theorem 4 and rule $\mathcal{R}4$, if $k-1$ receives $p_i$, $k-1$ delivers and forwards $p_i$. By Lemma 1 and the argument for case $(i)$, $k$ eventually receives $p_i$ via $k$.

Node mobility is also considered to show that if a node broadcasts a message, all nodes that remain reachable from $p$ receive the message, even if they are mobile and do not stay in the immediate neighbourhood of the sending node. First, a proof is presented to show that if a node $q$ moves away to $q'$, $q$ eventually receives the message sent by $p$. The case covers the first step of induction where $q$ moves such that there is one intermediate node between $p$ and $q$, as shown in Figure B.1 $(iii)$.

**case (iii)** By case $(i)$, $r$ receives $p_i$ and by Theorem 4, $r$ delivers $p_i$. Then by $\mathcal{R}4$, $r$ forwards $p_i$. The case then reduces to case $(i)$, where $q$ is a neighbour of $r$.

Finally, a proof is presented to show that if $q$ moves such that there is a sequence of nodes that have to forward $p_i$, as in Figure B.1 $(iv)$, $q$ eventually receives the message. The proof for this case is an induction on the series of nodes through which the message is forwarded through to reach $q$.

**case (iv)** Case $(iii)$ is the base case for the induction; the length of the series of intermediate nodes is 1. For the inductive step, it is assumed that $q$ moves to location shown as $q''$ as shown in Figure B.1 $(iv)$, and $r$ is replaced by a series of nodes of length $|l-1|$, such that the last node in the series is $s$. The proof for the inductive step when the length of the series is $l$, and $q$ moves to $q'$ such that the last node in the series is $t$ is shown in Figure B.1. Given the assumption for the series with length $l-1$, $s$ has received $p_i$. From Theorem 4 and rule $\mathcal{R}4$, $s$ delivers and forwards $p_i$. Again from the argument in case $(i)$, $t$ receives and delivers $p_i$ via $s$ and then $q'$ receives $p_i$ via $t$.

# Appendix C

# Pseudocode

This appendix shows the pseudocode for the protocols used in TransMAN and presented in Chapter 4. The pseudocode is broken into modules and each module presents a sub-protocol used by TransMAN. These modules are used only to break the presentation of the protocols into small components and are not representative of the implementation discussed in Chapter 5.

For ease of reference all the data structures used in the rest of the modules are presented at the outset in the first module. The rest of the modules are all referenced from Chapter 4 to illustrate the protocol.

**Module 1** Reliable Broadcast: Initialisation

1: $id \leftarrow \{\}$ // Node identifier
2: $last\_sent \leftarrow \{\}$ // Message sequence number Counter
3: $last\_delivered \leftarrow \{\}$ // The message last delivered
4: $last\_delivered\_from \leftarrow \{\}$ // List of messages last delivered from other nodes
5: $ReceivedMessages \leftarrow \{\}$ // List of messages received but not delivered
6: $DeliveredMessages \leftarrow \{\}$ // List of delivered messages
7: $DBG \leftarrow \emptyset$ // Delivered Before Graph
8: $PacketsToSend \leftarrow \{\}$ // Queue of messages pending transmission. Broadcast optimisers manipulate this queue.
9: Initialise SendThread // Generate timeout messages in absence of application messages
10: Initialise Stability // Message stability module
11: FD.$wait\_length \leftarrow 10$ // Failure detector module
12: $suspected \leftarrow \{\}$ // List of suspected nodes
13: Membership.$group\_view \leftarrow \{id\}$ // Current group view
14: Membership.$\mathcal{T} \leftarrow \emptyset$ // Tentative group view
15: Membership.$t\_views \leftarrow \emptyset$ // Sorted list of group view pending agreements

**Module 2** Reliable Broadcast: Sending Messages

```
 1: procedure SendApplicationMessage(msg)
 2:     msg ← CreateNewPacket("app")
 3:     Deliver(msg) // Alg. 4, line 11 on the following page
 4:     PacketsToSend.Enque(msg)
 5: end procedure

 6: procedure SendThread
 7:     msg ← PacketsToSend.Deque()
 8:     if msg =⊥ then
 9:         msg ← CreateNewPacket("timeout")
10:         Deliver(msg)
11:         UDP Broadcast msg
12:     end if
13: end procedure

14: procedure CreateNewPacket(type)
15:     msg ← new Packet
16:     msg.sender ← id
17:     msg.seqNo ← last_sent
18:     msg.type ← type
19:     msg.lastDelivered ← last_delivered
20:     msg.lastSent ← last_sent
21:     last_sent ← last_sent + 1
22: end procedure
```

**Module 3** Reliable Broadcast: Receiving Messages

```
 1: procedure OnRecv(msg)
 2:     if msg.type is nack then
 3:         RespondToNack(msg)
 4:     end if
 5:     Membership.Received(msg) // Inform membership layer
 6:     if msg ∉ DeliveredMessages ∧ (canDeliver(msg)) then
 7:         PacketsToSend.enque(msg) // Forward delivered message
 8:         Deliver(msg)
 9:         DeliverFromReceivedMessages()
10:     else
11:         ReceivedMessages.enque(msg)
12:         SendNacksFor(msg)
13:     end if
14: end procedure
```

**Module 4** Reliable Broadcast: Delivering Messages

```
1:  procedure CanDeliver(msg)
2:      return msg.lastDelivered = ⊥ ∨ (msg.lastDelivered ∈ DeliveredMessages ∧
            msg.lastSent ∈ DeliveredMessages)
3:  end procedure

4:  procedure DeliverFromReceivedMessages
5:      for all msg ∈ ReceivedMessages do
6:          if (CanDeliver( msg )) then
7:              Deliver( msg )
8:          end if
9:      end for
10: end procedure

11: procedure Deliver(msg)
12:     DeliveredMessages.Enque(msg)
13:     last_delivered ← msg
14:     DBG.AddEdge(msg.lastDelivered, msg)
15:     DBG.AddEdge(msg.lastSent, msg)
16:     // Call backs for various higher layers
17:     FD.Delivered(msg)
18:     Stability.Delivered(msg)
19:     Membership.Delivered(msg)
20:     last_delivered_from[msg.sender] ← msg
21: end procedure
```

**Module 5** Reliable Broadcast: Negative Acknowledgements

```
1:  procedure SendNacksFor(msg)
2:      if msg.lastDelivered ∉ DeliveredMessages∧ msg.lastDelivered ∉
            ReceivedMessages then
3:          nack ← CreateNackFor(msg.lastDelivered)
4:          PacketsToSend.enque(nack)
5:      else
6:          SendNacksFor(msg.lastDelivered) // Recursively look for missing dependency
7:      end if
8:      if msg.lastSent ∉ DeliveredMessages∧ msg.lastSent ∉ ReceivedMessages then
9:          nack ← CreateNackFor(msg.lastSent)
10:         PacketsToSend.Enque(nack)
11:     else
12:         SendNacksFor(msg.lastSent) // Recursively look for missing dependency
13:     end if
14: end procedure

15: procedure RespondToNack(msg)
16:     if msg ∈ DeliveredMessages or msg ∈ ReceivedMessages then
17:         PacketsToSend.Enque(msg)
18:     end if
19: end procedure

20: procedure CreateNackFor(sender, seqNo)
21:     nack ← new Packet
22:     nack.type ← NACK
23:     nack.sender ← sender
24:     nack.seqNo ← seqNo
25:     return nack
26: end procedure
```

**Module 6** Stability

```
1:  // Messages stabilised in topological sort order of the DBG. Lemma 8 on page 71
2:  procedure DELIVERED(msg)
3:      buf ← DBG.topological_sort
4:      for all m in buf do
5:          if IsStable?(msg) then
6:              delete msg from DBG and DeliveredMessages
7:          else
8:              return
9:          end if
10:     end for
11: end procedure

12: // OPD used for determining message delivery and thus stability
13: procedure ISSTABLE?(msg)
14:     for all mid ∈ last_delivered_from do
15:         if OPD(id, msg, mid) ≠ true then
16:             return false
17:         end if
18:     end for
19:     return true
20: end procedure

21: procedure OPD(node, m, n)
22:     return true if DBG has a path from m to n
23: end procedure
```

**Module 7** Failure Detector

```
1:  procedure DELIVERED(msg)
2:      for all node ∈ group_view do
3:          if FailureToReceive(node) ∨ FailureToBroadcast(node) then
4:              suspected ← node
5:          else
6:              delete node from suspected
7:          end if
8:      end for
9:  end procedure

10: procedure FAILURETORECEIVE(node)
11:     return OPD(node, [id, last_sent − wait_length], last_delivered_from[node])
12: end procedure

13: procedure FAILURETOBROADCAST(node)
14:     if Number of messages delivered since last message delivered from node >
            wait_length then
15:         return true
16:     else
17:         return false
18:     end if
19: end procedure
```

**Module 8** Membership - I

```
1:  procedure RECEIVED(msg)
            // First message from msg.sender
2:      if msg.sender ∉ group_view ∧ msg.sender ∉ 𝒯 then
3:          update 𝒯 to include msg.sender
4:          pkt ← CreateNewPacket("init-view-change")
5:          pkt.suggested_view ← 𝒯
6:          PacketsToSend.Enqueue(pkt)
7:          Deliver(msg) // incorporate msg.sender in the reliable broadcast
8:      end if
9:  end procedure

10: procedure DELIVERED(msg)
11:     if msg.type = "init-view-change" then
12:         𝒯_rcvd ← msg.suggested_view
13:         if 𝒯 < 𝒯_rcvd ∨ 𝒯 ⊂ 𝒯_rcvd then
14:             UpdateView(𝒯, 𝒯_rcvd) // Incorporate new nodes and higher seq_nos
15:             𝒯.seconded_by ← {}
16:             pkt ← CreateNewPacket("init-view-change")
17:             pkt.suggested_view ← 𝒯
18:             PacketsToSend.Enqueue(pkt)
19:         end if
20:         𝒯.seconded_by.Enqueue(msg)
21:     else
22:         for all t ∈ t_views.reversed do // First looking at the latest views
23:             reached ← UpdateAgreement(t, msg)
24:             if reached then
25:                 Membership.InstallView(t) // Install new view
26:             end if
27:         end for
28:     end if
29:     if suspected informs of node failure or recovery not yet reflected in 𝒯 then
30:         update 𝒯 to reflect changes in suspected
31:         pkt.suggested_view ← 𝒯
32:         PacketsToSend.Enqueue(pkt)
33:     end if
34:     if 𝒯.seconded_by.Includes(node) ∀ node ∈ 𝒯 then
35:         t_views.Enqueue(𝒯) // Initiate agreement on 𝒯
                    // if it has been proposed by all nodes in 𝒯
36:     end if
37: end procedure
```

**Module 9** Membership - II

```
1:  procedure UpdateAgreement(v, msg)
2:      for all node ∈ v do
3:          for all second ∈ v.seconded_by do
4:              if OPD(id, second, last_delivered_from[v]) = false then
5:                  return false
6:              end if
7:          end for
8:      end for
9:      return true
10: end procedure

11: procedure InstallView(v)
12:     group_view ← v
13:     T ← {}
14:     for all m ∈ DeliveredMessages do
15:         Stability.Delivered(m) // Stabilise messages delivered during view install
16:     end for
17:     delete v from t_views if v < t_view // Terminate preceding views
18: end procedure
```

# Bibliography


Abbadi, A. E., Skeen, D. & Cristian, F. (1985). An efficient, fault-tolerant protocol for replicated data management, *PODS '85: Proceedings of the fourth ACM SIGACT-SIGMOD symposium on Principles of database systems*, ACM Press, New York, NY, USA, pp. 215–229.

Agarwal, M., Cho, J. H., Gao, L. & Wu, J. (2004). Energy efficient broadcast in wireless ad hoc networks with hitch-hiking, *INFOCOM*.

Amir, Y., Dolev, D., Kramer, S. & Malki, D. (1992). Membership algorithms for multicast communication groups, *Workshop on Distributed Algorithms (WDAG-6)*, number 647 in *LNCS*, Springer, Haifa, Israel, pp. 292–312.

Amir, Y., Moser, L. E., Melliar-Smith, P. M., Agarwal, D. A. & Ciarfella, P. (1995). The Totem single-ring ordering and membership protocol, *ACM Transactions on Computer Systems* **13**(4): 311–342.

Andel, T. & Yasinsac, A. (2006). On the credibility of manet simulations, **39**(7): 48–54.

Babaoğlu, O., Davoli, R. & Montresor, A. (1998). Group communication in partitionable systems: Specification and algorithms, *Technical Report UBLCS 98-01*, Department of Computer Science University of Bologna.

Babaoğlu, O., Davoli, R. & Montresor, A. (2001). Group membership in partitionable systems: Specification and algorithms, *IEEE Transactions on Software Engineering* **27**(4).

Babaoğlu, O., Davoli, R., Montresor, A. & Segala, R. (1998). System support for partition-aware network applications, *Proceedings of the 18th International Conference on Distributed Computing Systems*, Vol. 18, IEEE, pp. 184–191.

Babaoğlu, O. & Schiper, A. (1995a). On group communication in large-scale distributed systems, *ACM SIGOPS Operating Systems Review* **29**(1): 62–67.

Babaoğlu, O. & Schiper, A. (1995b). On group communication in large-scale distributed systems, *ACM Operating Systems Review* **29**(1): 62–67.

Bar-Yossef, Z., Friedman, R. & Kliot, G. (2006). Rawms -: random walk based lightweight membership service for wireless ad hoc network, *MobiHoc '06: Proceedings of the seventh ACM international symposium on Mobile ad hoc networking and computing*, ACM Press, New York, NY, USA, pp. 238–249.

Barber, M. N. & Ninham, B. W. (1970). *Random and Restricted Walks: Theory and Applications*, Gordon and Breach.

Barrett, C. L., Marathe, M. V., Engelhart, D. C. & Sivasubramaniam, A. (2002). Analyzing the short-term fairness of ieee 802.11 in wireless multi-hop radio networks, *10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'02)*, pp. 137 – 144.

Barron, P., Weber, S., Clarke, S. & Cahill, V. (2005). Experiences deploying an ad-hoc network in an urban environment, *in* J. Crowcroft, M. Conti & A. Passarella (eds), *IEEE ICPS Workshop on Multi-hop Ad hoc Networks: from theory to reality*, IEEE Computer Society, pp. 103–110.

Bensaou, B., Wang, Y. & Ko, C. C. (2000). Fair medium access in 802.11 based wireless ad-hoc networks, *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, IEEE Press, pp. 99–106.

Birman, K., Constable, R., Hayden, M., Kreitz, C., Rodeh, O., van Renesse, R. & Vogels, W. (2000). The horus and ensemble projects: Accomplishments and limitations, *Proc.*

*of the DARPA Information Survivability Conference and Exposition (DISCEX '00)*, Hilton Head, South Carolina.

Birman, K. & Joseph, T. (1987a). Exploiting virtual synchrony in distributed systems, *Proceedings of the eleventh ACM Symposium on Operating systems principles*, ACM Press, pp. 123–138.

Birman, K. P. (1993). The process group approach to reliable distributed computing, *Commun. ACM* **36**(12): 37–53.

Birman, K. P., Hayden, M., Ozkasap, O., Xiao, Z., Budiu, M. & Minsky, Y. (1999). Bimodal multicast, *ACM Transactions on Computer Systems* **17**(2): 41–88.

Birman, K. P. & Joseph, T. A. (1987b). Reliable communication in the presence of failures, *Transactions on Computer Systems* **5**(1): 47–76.

Birman, K. P., van Renesse, R. & Vogels, W. (2001). Spinglass: Secure and scalable communications tools for mission-critical computing, *International Survivability Conference and Exposition. DARPA DISCEX-2001*, Anaheim, California.

Bommaiah, E., Liu, M., McAuley, A. & Talpade, R. (1998). Amroute: Ad-hoc multicast routing protocol, Internet-Draft draft-talpade-manet-amroute-00.txt.

Boppana, R. & Zheng, Z. (2005). On the performance of metro-scale ad hoc networks, *Wireless Communications and Networking Conference*, Vol. 3, IEEE, pp. 1453–1458.

Breslau, L., Estrin, D., Fall, K., Floyd, S., Heidemann, J., Helmy, A., Huang, P., McCanne, S., Varadhan, K., Xu, Y. & Yu, H. (2000). Advances in network simulation, *IEEE Computer* **33**(5): 59–67.

Briesemeister, L. & Hommel, G. (2002). Localized group membership service for ad hoc networks, *Proceedings of International Workshop on Ad Hoc Networking (IWAHN)*, pp. 94–100.

Cahen, D., Sasson, Y. & Schiper, A. (2002). On the accuracy of manet simulators, *Proceedings of Mobile Computing*.

Chandra, R., Ramasubramanian, V. & Birman, K. (2001). Anonymous gossip: Improving multicast reliability in ad-hoc networks, *International Conference on Distributed Computing Systems (ICDCS 2001)*, Phoenix, Arizona.

Chandra, T. D., Hadzilacos, V. & Toueg, S. (1996). The weakest failure detector for solving consensus, *J. ACM* **43**(4): 685–722.

Chandra, T. D. & Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems, *Journal of the ACM* **43**(2): 225–267.

Chen, B., Jamieson, K., Balakrishnan, H. & Morris, R. (2001). Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks, *Proceedings of the Seventh ACM International Conference on Mobile Computing and Networking*, pp. 85–96.

Cheriton, D. R. & Zwaenepoel, W. (1985). Distributed process groups in the v kernel, *ACM Trans. Comput. Syst.* **3**(2): 77–107.

Chlebus, B. S., Gasieniec, L., Gibbons, A., Pelc, A. & Rytter, W. (2000). Deterministic broadcasting in ad hoc radio networks, *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 861–870.

Chlebus, B. S., Gasieniec, L., Lingas, A. & Pagourtzis, A. T. (2001). Oblivious gossiping in ad-hoc radio networks, *Proceedings of the 5th international workshop on Discrete algorithms and methods for mobile computing and communications*, ACM Press, pp. 44–51.

Chockler, G. V., Keidar, I. & Vitenberg, R. (2001). Group communication specifications: A comprehensive study, *ACM Computing Surveys(CSUR)* **33**(4): 427–469.

Cristian, F. (1991). Reaching agreement on processor-group membership in synchronous distributed systems, *Distributed Computing* **4**(4): 175–188.

Cristian, F. & Fetzer, C. (1999). The timed asynchronous distributed system model, *IEEE Transactions on Parallel and Distributed Systems* **10**(6): 642–657.

Cristian, F. & Schmuck, F. (1995). Agreeing on processor group membership in timed asynchronous distributed systems, *Technical Report CSE95-428*, Computer Science Department, University of California, San Diego.

Dai, F. & Wu, J. (2003). Distributed dominant pruning in ad hoc wireless networks, *Proceedings of ICC*, Vol. 1.

Dai, F. & Wu, J. (2004). Performance analysis of broadcast protocols in ad hoc networks based on self-pruning, *IEEE Trans. Parallel Distrib. Syst.* **15**(11): 1027–1040.

Das, A., Gupta, I. & Motivala, A. (2002). Swim: Scalable weakly-consistent infection-style process group membership protocol, *Proceedings of the International Conference on Dependable Systems and Networks 2002 (DSN 2002)*, pp. 303–312.

Deering, S. (1989). Host extensions for ip multicasting, IETF RCF 1112.

Dessmark, A. & Pelc, A. (2001). Deterministic radio broadcasting at low cost, *STACS '01: Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science*, Springer-Verlag, pp. 158–169.

Dolev, D., Kramer, S. & Malki, D. (1992). Total ordering of messages in broadcast domains, *Technical Report CS92-9*, Computer Science Department, The Hebrew University of Jerusalem.

Dolev, D. & Malki, D. (1995). The design of the transis system, Dagstuhl Workshop on Unifying Theory and Practice in Distributed Computing.

Dolev, D., Malki, D. & Strong, R. (1994). *An Asynchronous Membership Protocol that Tolerates Partitions*, IBM Almaden Research Center.

Dolev, S. (2000). *Motivating Self-Stabilization*, MIT Press.

Dolev, S., Gilbert, S., Lynch, N. A., Shvartsman, A. A. & Welch, J. L. (2003). Geoquorums: Implementing atomic memory in mobile ad hoc networks, *in* F. E. Fich (ed.), *Distributed algorithms*, Vol. 2848, pp. 306–320.

Dolev, S., Schiller, E. & Welch, J. (2006). Random walk for self-stabilizing group communication in ad-hoc networks, *IEEE Transactions on Mobile Computing* **5**.

Eriksson, J., Faloutsos, M. & Krishnamurthy, S. (2004). Scalable ad hoc routing: The case for dynamic addressing, *INFOCOM*.

Eugster, P. T., Guerraoui, R., Handurukande, S. B., Kouznetsov, P. & Kermarrec, A.-M. (2003). Lightweight probabilistic broadcast, *ACM Trans. Comput. Syst.* **21**(4): 341–374.

Ezhilchelvan, P., Macedo, R. & Shrivastava, S. (1995). Newtop: A fault-tolerant group communication protocol, *ICDCS-15*, Vancouver.

Fekete, A., Lynch, N. & Shvartsman, A. (2001). Specifying and using a partitionable group communication service, *ACM Transactions on Computer Systems* **19**(2): 171–216.

Fischer, M. J., Lynch, N. A. & Paterson, M. S. (1985). Impossibility of distributed consensus with one faulty process, *Journal of the ACM (JACM)* **32**(2): 374–382.

Foroozan, F. & Tepe, K. (2005). A high performance cluster-based broadcasting algorithm for wireless ad hoc networks based on a novel gateway selection approach, *PE-WASUN '05: Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, ACM Press, New York, NY, USA, pp. 65–70.

Friedman, R. (2003). Fuzzy group membership, *Future Directions in Distributed Computing*, Vol. 2584 of *Lecture Notes in Computer Science*, Springer, pp. 114–118.

Friedman, R. & Tcharny, G. (2003a). Fuzzy membership based reliable delivery for mobile ad-hoc networks, *Technical Report CS-2003-14*, Technion - Israel Institute of Technology.

Friedman, R. & Tcharny, G. (2003b). Stability detection in mobile ad-hoc networks, *Technical report*, Israel Institute of Technology.

Friedman, R. & van Renesse, R. (1996). Strong and weak virtual synchrony in horus, *15th Symposium on Reliable Distributed Systems (SRDS '96)*.

Gaber, I. & Mansour, Y. (1995). Broadcast in radio networks, *SODA: ACM-SIAM Symposium on Discrete Algorithms*.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional Computing Series.

Garcia-Luna-Aceves, J. J. & Madruga, E. (1999). A multicast routing protocol for ad-hoc networks, *Proceedings of IEEE INFOCOM*, pp. 784–792.

Gasieniec, L. & Lingas, A. (2002). On adaptive deterministic gossiping in ad hoc radio networks, *Inf. Process. Lett.* **83**(2): 89–93.

Gavidia, D., Voulgaris, S. & van Steen, M. (2005). Epidemic-style monitoring in large-scale wireless sensor networks, *Technical Report IR-CS-012*, Vrije Universiteit Amsterdam.

Gopalswamy, T., Singhal, M., Panda, D. & Sadayappan, P. (2002). A reliable multicast algorithm for mobile ad hoc networks, *Proceedings of the 22nd International Conference on Distributed Computing Systems*.

Gupta, I., Chandra, T. D. & Goldszmidt, G. S. (2001). On scalable and efficient distributed failure detectors, *PODC '01: Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*, ACM Press, New York, NY, USA, pp. 170–179.

Gupta, P. & Kumar, P. (2000). The capacity of wireless networks, *IEEE Transaction on Information Theory* **46**(2).

Haas, Z. & Pearlman, M. (1999). The zone routing protocol (zrp) for ad hoc networks, Internet Draft, draft-ietf-manet-zone-zrp-02.txt.

He, T., Stankovic, J. A., Lu, C. & Abdelzaher, T. (2003). Speed: A stateless protocol for real-time communication in sensor networks, *23rd International Conference on Distributed Computing Systems*, IEEE, p. 46.

Hiltunen, M. & Schlichting, R. (1995). Properties of membership services.

Jacquet, P., Muhlethaler, P., Qayyum, A., Laouiti, A., Viennot, L. & Clausen, T. (2003). Optimized link state routing protocol (olsr), RFC 3626.

Ji, L. & Corson, M. (1998). A lightweight adaptive multicast algorithm, *Global Telecommunications Conference*, Vol. 2, pp. 1036–1042.

Ji, L. & Corson, M. S. (2001). Differential destination multicast - a MANET multicast routing protocol for small groups, *INFOCOM*, pp. 1192–1202.

Jo, J., Eugster, P. T. & Hubaux, J. (2003). Route driven gossip: Probabilistic reliable multicast in ad hoc networks, *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 3, pp. 2229 – 2239.

Juan-Carlos Cano, P. M. (2001). Evaluating the energy-consumption reduction in a manet by dynamically switching-off network interfaces, *ISCC '01: Proceedings of the Sixth IEEE Symposium on Computers and Communications (ISCC'01)*, IEEE Computer Society, Washington, DC, USA, p. 186.

Kaashoek, M., Tanenbaum, A. & Verstoep, K. (1993). Group communication in amoeba and its applications, *Distributed Systems Engineering Journal* pp. 48–58.

Keidar, I., Sussman, J., Marzullo, K. & Dolev, D. (2002). Moshe: A group membership service for wans, *ACM Trans. Comput. Syst.* **20**(3): 191–238.

Ko, Y. & Vaidya, N. (1998). Geocasting in mobile ad hoc networks: Location-based multicast algorithms.

Kunz, T. (2003). Multicasting in mobile ad-hoc networks: achieving high packet delivery ratios, *Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research*, IBM Press, pp. 156–170.

Kurkowski, S., Camp, T. & Colagrosso, M. (2005). Manet simulation studies: the incredibles, *SIGMOBILE Mob. Comput. Commun. Rev.* **9**(4): 50–61.

la Kloul, L. & Valois, F. (2005). Investigating unfairness scenarios in manet using 802.11b, *PE-WASUN '05: Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, ACM Press, New York, NY, USA, pp. 1–8.

Lee, S.-J., Su, W. & Gerla, M. (2000). On-demand multicast routing protocol (odmrp) for ad hoc networks, *Internet Draft, draft-ietf-manet-odmrp-02.txt*.

Lee, S.-J., Su, W. & Gerla, M. (2001). Wireless ad hoc multicast routing with mobility prediction, *Mob. Netw. Appl.* **6**(4): 351–360.

Li, X.-Y., Song, W.-Z. & Wang, W. (2005). A unified energy-efficient topology for unicast and broadcast, *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, ACM Press, New York, NY, USA, pp. 1–15.

Liang, W. (2002). Constructing minimum-energy broadcast trees in wireless ad hoc networks, *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, ACM Press, New York, NY, USA, pp. 112–122.

Lim, H. & Kim, C. (2001). Flooding in wireless ad hoc networks, *Computer Communication Journal* **24**(2-3): 353–363.

Lin, C. R. & Gerla, M. (1997). Adaptive clustering for mobile wireless networks, *IEEE Journal of Selected Areas in Communications* **15**(7): 1265–1275.

Liu, J., Liu, J., Reich, J., Cheung, P. & Zhao, F. (2004). Distributed group management in sensor networks: Algorithms and applications to localization and tracking., *Telecommunication Systems* **26**(2-4): 235–251.

Liu, J., Sacchetti, D., Sailhan, F. & Issarny, V. (2005). Group management for mobile ad hoc networks: design, implementation and experiment, *MDM '05: Proceedings of the 6th international conference on Mobile data management*, ACM Press, New York, NY, USA, pp. 192–199.

Lou, W. & Wu, J. (2002). On reducing broadcast redundancy in ad hoc wireless networks, *IEEE Transactions on Mobile Computing* **1**(2): 111– 122.

Lou, W. & Wu, J. (2004). Double-covered broadcast(dcb): A simple reliable broadcast algorithm in manets, *INFOCOM*.

Luo, J., Eugster, P. T. & Hubaux, J.-P. (2004). Pilot: Probabilistic lightweight group communication system for ad hoc networks, *IEEE Transactions on Mobile Computing* **3**(2).

Lynch, N. & Tuttle, M. (1989). An introduction to input/output automata, *CWI-Quarterly* **2**(3): 219–246.

Malloth, C. P. & Schiper, A. (1995). View synchronous communication in large scale distributed systems, *Proceedings of the 2nd Open Workshop of the ESPRIT project BROADCAST (#6360)*, Grenoble, France.

Meier, R., Killijian, M.-O., Cunningham, R. & Cahill, V. (2001). Towards proximity group communication, *Principles of Mobile Computing*.

Meling, H., Montresor, A., Babaoglu, O. & Helvik, B. E. (2002). Jgroup/arm: A distributed object group platform with autonomous replication management for dependable computing, *Technical Report UBLCS 2002-12*, University of Bologna, Italy.

Melliar-Smith, P. & Moser, L. (1993). Trans: a reliable broadcast protocol, **140**(6): 481–493.

Metcalfe, R. M. & Boggs, D. R. (1976). Ethernet: distributed packet switching for local computer networks, *Commun. ACM* **19**(7): 395–404.

Miranda, H., Pinto, A. & Rodrigues, L. (2001). Appia, a flexible protocol kernel supporting multiple coordinated channels, *Proceedings of the 21st International Conference on Distributed Computing Systems*, IEEE, Phoenix, Arizona, pp. 707–710.

Mishra, S., Fetzer, C. & Cristian, F. (2002). The timewheel group communication system, *IEEE Trans. Comput.* **51**(8): 883–899.

Mishra, S., Peterson, L. L. & Schlichting, R. D. (1993). Consul: a communication substrate for fault-tolerant distributed programs, *Distributed Systems Engineering* **1**(2): 87–103.

Moser, L. E., Amir, Y., Melliar-Smith, P. M. & Agarwal, D. A. (1994). Extended virtual synchrony, *The 14th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp. 56–65.

Moser, L. E., Melliar-Smith, P. & Agarwal, V. (1994). Processor membership in asynchronous distributed systems, *IEEE Transaction on Parallel and Distributed Systems* **5**(5): 459–473.

Mullender, S. (ed.) (1989). *Distributed systems*, ACM Press, New York, NY, USA.

Orecchia, L., Panconesi, A., Petrioli, C. & Vitaletti, A. (2004). Localized techniques for broadcasting in wireless sensor networks, *DIALM-POMC '04: Proceedings of the 2004 joint workshop on Foundations of mobile computing*, ACM Press, New York, NY, USA, pp. 41–51.

Özalp Babaoğlu, Davoli, R., Giachini, L.-A. & Baker, M. (1995). Relacs: A communication infrastructure for constructing reliable applications in large-scale distributed systems, *Technical Report UBLCS-94-15*, Laboratory of Computer Science, University of Bologna.

Pagani, E. & Rossi, G. P. (1997). Reliable broadcast in mobile multihop packet networks, *MobiCom '97: Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking*, ACM Press, New York, NY, USA, pp. 34–42.

Peng, W. & Lu, X.-C. (2000). On the reduction of broadcast redundancy in mobile ad hoc networks, *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, IEEE Press, pp. 129–130.

Perkins, C. E., Belding-Royer, E. M. & Das, S. (2003). Ad hoc on demand distance vector (aodv) routing, IETF RFC 3561.

Perkins, C. E. (ed.) (2001). *DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks*, Addison-Wesley, chapter 5, pp. 139–172.

Peterson, L. L., Buchholz, N. C. & Schlichting, D. (1989). Preserving and using context information in interprocess communications, *Transactions on Computer Systems* **7**(3): 217–246.

Peterson, L. L. & Davie, B. S. (2003). *Computer Networks*, Morgam Kaufmann.

Pradhan, P. & Helal, S. (2003). An efficient algorithm for maintaining consistent group membership in ad hoc networks, Proceedings of the First International Workshop on Mobile Distributed Computing (MDC'03).

Qayyum, A., Viennot, L. & Laouiti, A. (2002). Multipoint relaying for flooding broadcast messages in mobile wireless networks, *hicss* **09**: 298.

Roman, G.-C., Huang, Q. & Hazemi, A. (2001). Consistent group membership in ad hoc networks, *Proceedings of the 23rd international conference on Software engineering*, IEEE Computer Society, pp. 381–388.

Royer, E. M. & Perkins, C. E. (1999). Multicast operation of the ad hoc on-demand distance vector routing protocol, *Proceedings of MobiCom '99*, pp. 207–218.

Sahimi, M. (1994). *Applications of Percolation Theory*, Taylor and Francis.

Sasson, Y., Cavin, D. & Schiper, A. (2003). Probabilistic broadcast for flooding in wireless mobile ad hoc networks, *Proceedings of IEEE Wireless Communications and Networking Conference*, Vol. 2, pp. 1124 – 1130.

Schiper, A., Birman, K. & Stephenson, P. (1991). Lightweight causal and atomic group multicast, *ACM Transactions on Computer Systems (TOCS)* **9**(3): 272–314.

Schiper, A. & Ricciardi, A. (1993). Virtually-synchronous communication based on a weak failure suspector, *Technical report*, LSR, EPFL.

Schneider, F. B. (1990). Implementing fault-tolerant services using the state machine approach: a tutorial, *ACM Comput. Surv.* **22**(4): 299–319.

Singh, K. (2006). On group communication, congestion avoidance and node starvation in manets. Third Minema Workshop.

Singh, K., Nedos, A., Gaertner, G. & Clarke, S. (2005). Message stability and reliable broadcasts in mobile ad-hoc networks, *in* V. Syrotiuk & E. Chávez (eds), *ADHOC-NOW*, LNCS 3788, pp. 297–310.

Stojmenovic, I., Seddigh, M. & Zunic, J. (2002). Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks, *IEEE Transactions on Parallel and Distributed Systems* **13**(1).

Sucec, J. & Marsic, I. (2000). An efficient distributed network-wide broadcast algorithm for mobile ad hoc networks, *Technical report*, Rutgers University.

Sun, M., Feng, W. & Lai, T. H. (2001). Location aided broadcast in wireless ad hoc networks, *Proceedings of IEEE Global Telecommunications Conference*, Vol. 5, pp. 2842 – 2846.

Sussman, J., Keidar, I. & Marzullo, K. (2000). Optimistic virtual synchrony, *SRDS '00: Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS'00)*, IEEE Computer Society, Washington, DC, USA, p. 42.

Tang, K., Obraczka, K., Lee, S.-J. & Gerla, M. (2002). Congestion controlled adaptive lightweight multicast in wireless mobile ad hoc networks., *Proceedings of the Seventh IEEE Symposium on Computers and Communications (ISCC)*, pp. 967–972.

Tseng, Y.-C., Ni, S.-Y., & Shih, E.-Y. (2001). Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network, *ICDCS '01: Proceedings of the The 21st International Conference on Distributed Computing Systems*, IEEE Computer Society, Washington, DC, USA, p. 481.

Tseng, Y.-C., Ni, S.-Y., Chen, Y.-S. & Sheu, J.-P. (2002). The broadcast storm problem in a mobile ad hoc network, *Wirel. Netw.* **8**(2/3): 153–167.

Uchida, J., Chen, W. & Wada, K. (2004). Acknowledged broadcasting and gossiping in ad hoc radio networks, *Lecture Notes in Computer Science*, Vol. 3144, pp. 223–234.

van Renesse, R., Maffeis, S. & Birman, K. (1996). Horus: A flexible group communications system, *Communications of the ACM* **39**(4): 76–83.

Vieira, M. S. & Rosa, N. S. (2005). A reconfigurable group management middleware service for wireless sensor networks, *MPAC '05: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, ACM Press, New York, NY, USA, pp. 1–8.

Viswanath, K. & Obraczka, K. (2002). An adaptive approach to group communications in multi-hop ad hoc networks, *International Conference on Networking*.

Vollset, E. W. & Ezhilchelvan, P. D. (2005). Design and performance-study of crash-tolerant protocols for broadcasting and reaching consensus in manets, *SRDS '05: Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems (SRDS'05)*, IEEE Computer Society, Washington, DC, USA, pp. 166–178.

Wan, C.-Y. & Campbell, A. T. (2002). Psfq: A reliable transport protocol for wireless sensor networks, Proc. of First ACM International Workshop on Wireless Sensor Networks and Applications.

Williams, B. & Camp, T. (2002). Comparison of broadcasting techniques for mobile ad hoc networks, *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, ACM Press, New York, NY, USA, pp. 194–205.

Wu, J. & Dai, F. (2003). Broadcasting in ad hoc networks based on self-pruning, *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 3, IEEE, pp. 2240 – 2250.

Wu, J. & Dai, F. (2004). A generic distributed broadcast scheme in ad hoc wireless networks, *IEEE Transactions on Computers* **53**(10): 1343–1354.

Wu, J. & Dai, F. (2005). Efficient broadcasting with guaranteed coverage in mobile ad hoc networks, *IEEE Transactions on Mobile Computing* **4**(3): 259–270.

Wu, J. & Li, H. (1999). On calculating connected dominating set for efficient routing in ad hoc wireless networks, *In Proceedings of the Third International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication (DIALM)*.

Ye, F., Luo, H., Cheng, J., Lu, S. & Zhang, L. (2002). A two-tier data dissemination model for large-scale wireless sensor networks, *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, ACM Press, New York, NY, USA, pp. 148–159.

Yi, Y., Gerla, M. & Obraczka, K. (2003). Scalable team multicast in wireless networks exploiting coordinated motion, *Ad Hoc Networks Journal* .