Delivering Messages in Disconnected Mobile Ad-Hoc Networks

by

Ritesh Shah

B.Engg., Rajiv Gandhi Technical University, 2001

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming
to the required standard

The University of British Columbia

March 2004

© Ritesh Shah, 2004

Library Authorization

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Ritesh Shah

Name of Author (*please print*)

08/03/2004

Date (dd/mm/yyyy)

Title of Thesis: Delivering Messages in Disconnected Mobile Ad-Hoc Networks

Degree: Master of Science

Year: 2004

Department of Computer Science

The University of British Columbia

Vancouver, BC Canada

Abstract

Rapid advancements in wireless technologies have fuelled parallel research in two areas, one in which wireless devices use a central authority to establish and maintain the network and one in which no such central authority is used. Mobile Ad-Hoc networks (MANETs) are part of the latter research. Most researchers define a mobile ad-hoc network as “a self-organizing network formed on the fly without the aid of any established infrastructure or central authority”.

Many routing protocols have been developed to establish and maintain routes in MANETs. They try to address the unique challenges that MANETs present over traditional wired networks. Some of these challenges are use of unreliable wireless medium for communication and lack of a central authority to facilitate any communication in the network. These protocols find a route to a destination if such a route exists. However, in the wireless medium, links are susceptible to frequent failures, which can cause partitions in the network. Current routing protocols use a passive delivery approach for packets destined to a host in the other partition. Packets destined to a disconnected host are dropped after some route repair attempts. This thesis presents a novel protocol, *Voilà*, that delivers messages between disconnected hosts. *Voilà* uses the nodes moving between the neighbourhoods of the source and destination nodes to act as carriers of messages. It uses a novel *Carrier Select* algorithm to select carrier nodes in the source partition. The thesis describes the protocol in detail and provides a simulation-based evaluation of its performance compared to other possible schemes and the optimal scheme.

Contents

Abstract	ii
Contents	iii
List of Tables	v
List of Figures	vi
Acknowledgements	viii
Dedication	ix
1 Introduction	1
2 Background	5
2.1 802.11	7
2.2 AODV	11
2.3 DSR	13
2.4 DSDV	14
2.5 OLSR	15
2.6 ZRP	17
3 Related Work	19

4	Algorithm	23
4.1	The Source Node	23
4.2	Other Selected Nodes	26
4.3	Message Delivery	29
4.4	Neighbour List Request Timeout	30
5	Design	31
6	Evaluation and Results	36
6.1	Routing Protocol Parameters	36
6.2	Mobility Model	37
6.2.1	Random Way Point Model	37
6.2.2	Modifications in Random Way Point Model	38
6.3	Results	39
6.3.1	Source-only Protocol	39
6.3.2	Oracle Protocol	39
6.3.3	Voilà-GPS	40
6.3.4	Random Selection	41
6.4	Discussion	44
7	Future Work	55
8	Conclusion	57
	Bibliography	58

List of Tables

6.1	AODV parameter values used in the simulations	37
6.2	Parameter values kept constant throughout all experiments	47
6.3	Parameter values used in the density experiment	47

List of Figures

2.1	Hidden Terminal Problem, Nodes A and C are hidden from each other	10
4.1	(a) A network partition of 10 nodes (b) Node positions after some mobility	28
5.1	High level architecture of Voilà	34
5.2	System architecture with Voilà layer	35
5.3	Header added by Voilà layer	35
6.1	All six neighbours selected as carrier nodes in Voilà-basic	42
6.2	Angle between the velocity vectors of nodes 1 and 2 is 90 degrees . .	42
6.3	Selection of Carrier nodes when RANDOM_CONST is 50%	43
6.4	A typical snapshot of positions of nodes in the simulated region of 3000m x 600m	47
6.5	Number of messages delivered as density of nodes is varied	48
6.6	Number of messages delivered as density of nodes is varied (without Oracle protocol)	49
6.7	Percentage of undelivered messages when no aggressive delivery mechanism is used	49
6.8	Number of messages delivered as FIND_DST_INTERVAL is varied .	50
6.9	Number of messages delivered as FIND_DST_INTERVAL is varied (without Oracle protocol)	51

6.10 Percentage of undelivered messages carried by each protocol to the destinations	51
6.11 Number of messages delivered as MAX_HOLD_TIME is varied . . .	52
6.12 Number of messages delivered as MAX_HOLD_TIME is varied (without Oracle protocol)	53
6.13 Number of messages delivered as DEGREE_CONST is varied	53
6.14 Number of messages delivered as RANDOM_CONST is varied	54

Acknowledgements

First of all, I would like to thank my supervisor, Dr. Norm Hutchinson for his constant encouragement and support throughout this thesis. Without his insight and direction this research would not have made it this far. I would also like to thank Dr. Will Evans for supporting me financially through out the course of this thesis. Working with him has been a tremendous learning experience for me which I will cherish for the rest of my life. I would also like to thank him for his comments on this thesis and for refining the algorithm described in Chapter 4 of this thesis. I am thankful to Ken, Dima, Alex, Geoffery, Joseph, James and Suprio for their help and for creating such a warm atmosphere in the DSG lab. My special thanks goes to Chamath for being a pillar of strength and encouragement throughout my stay in Vancouver. Last but not the least, my thanks and gratitude to my parents and my sister, Sonal, who have always believed in me and my decisions. Their tremendous support made it possible for me to come to Vancouver and earn a Master's degree.

RITEESH SHAH

The University of British Columbia
March 2004

To my loving Grandmother

Chapter 1

Introduction

An ad-hoc network is a self-starting network formed on-the-fly by a group of mobile nodes without the aid of any centralized administration or established infrastructure. Ad-hoc networks find their use in situations where no fixed infrastructure is available or has been damaged by natural or man-made disaster.

Rapid advancements in wireless technology and increasingly affordable prices of wireless devices have made ad-hoc networks a reality. This has fuelled a lot of research activity in the field. Several protocols [RT99], [BMJHJ98], [PRDM01] have been developed to find and maintain routes between the nodes of an ad-hoc network. These routing protocols can be divided into four broad categories. First are the proactive protocols that use periodic advertisements to broadcast routing information, such as DSDV [PB94] and OLSR [Rfc26]. Second are on-demand protocols that search for routes on-demand, such as AODV [PR99] and DSR [JM96]. Third are those routing protocols that use a hybrid approach such as ZRP [HP97], which is a combination of the first two approaches. And fourth are those that perform hierarchical routing, which is similar to subnet routing in IP networks, such as MOCDS [AWF02]. While each approach has its advantages and disadvantages in finding a route between two mobile hosts when one exists, none of them deal with the case of

message delivery between two disconnected hosts very well.

Links in a MANET (Mobile Ad-hoc NETwork) are susceptible to frequent failures due to movement of nodes. This may cause some nodes to get disconnected from others. A message destined to such a disconnected node results in a delivery failure irrespective of the routing protocol used. Different protocols handle this situation differently but at most they invalidate the route after making one or more route repair attempts, if there was one already in use, and inform the source about the situation. In this case the source has to wait for the destination to come in its radio range (or the radio range of one of the connected nodes) again before it can deliver any message. Instead of following such a passive approach a more aggressive approach can be pursued.

Why is the question of message delivery between disconnected hosts so important as to warrant an aggressive approach? Consider a disaster relief scenario. Relief workers are working on several sites scattered in an area. The workers have mobile devices to communicate among themselves. The sites may be separated by a distance that is several times the radio range of the devices. In such a situation some of the sites might be disconnected from each other, forming multiple partitioned mobile ad-hoc networks in the area. While node movements between the sites may offer connectivity, it may be sporadic and for brief periods of time. In such a situation it would be helpful to have some mechanism in place for delivering messages between disconnected hosts.

Consider a similar scenario on a long beach having several scenic-spots separated from each other by a distance several times the radio range. While there are nodes moving between them, the scenic-spots may be disconnected for the majority of the time. One way to communicate across such partitioned ad-hoc networks is to

use nodes moving between them as carriers of messages. Nodes leaving network A and joining network B could carry messages from other nodes in A that are destined for nodes in B.

Now the question is how to select such a carrier node. One option is to select all the nodes in the connected graph containing the source as carrier nodes. This could result in unnecessary replication of messages and a waste of network bandwidth. So the goal is to find the “right carrier node” — the one that will come in contact with the destination within a certain period of time in the future. It is not possible for a source to choose such a “right carrier node” without the knowledge of the trajectories of all the nodes in the region (including the ones that are not in its connected subgraph). So a more refined goal could be to select those nodes as carrier nodes that have a higher probability of connecting to the destination in the future. One way to compute this probability is to use past association of nodes. Here past association of a node means those nodes that have come in contact with it within a certain period of time in the past. This could be a useful metric when nodes show a repetitive movement pattern, so the past movements of a node predict its future movements and hence its past associations predict its future associations. But this metric is not very useful in the scenarios where nodes gather occasionally like a beach or a disaster relief site. Therefore our goal is to have the source node select carrier nodes in every direction (as the position and direction of movement of the disconnected destination is not known) and to have the message propagate to the boundary of the partition and to minimize the redundancy in doing so. In this thesis we propose a completely decentralized protocol, *Voilà*, that replicates an undelivered message in the source partition using the *Carrier Select* algorithm.

Chapter 2 describes 802.11 technology and the routing protocols developed for MANETs. Chapter 3 looks at approaches proposed by other researchers to

address the same problem. Chapter 5 presents the high level design of our system and how it fits into the overall system design. In Chapter 4 we describe our *Carrier Select* algorithm in detail. Chapter 6 discusses the set up of the experiments and performance results of our protocol comparing them with the performance results of other possible schemes. We present future work in Chapter 7 and Chapter 8 concludes.

Chapter 2

Background

Nodes in MANETs communicate through the wireless medium. The wireless medium poses several challenges not encountered in wired media such as:

1. **Limited range.** The range of the radio signal sent by a wireless device is limited. It can be deciphered correctly by receivers only when they are within a certain distance of the source. This is due to power attenuation encountered by the radio signals as they travel through the medium.
2. **Limited bandwidth.** The bandwidth available in the wireless medium is limited and shared among the devices in the vicinity.
3. **Limited Power.** Most wireless devices are battery powered and therefore have limited power.
4. **Unreliable medium.** The communication in wired medium is inherently reliable. Rarely does data get lost due to low power of the signal or interference from external factors. This is not the case in wireless media where data can get lost due to interference from atmosphere or other devices.
5. **Less Secure.** As radio waves propagate through open medium, it is easier for a malicious node to eavesdrop on a communication.

In addition, wireless networks may encounter frequent change in topology due to node movement. MANETs pose an additional challenge in the sense that there is no centralized administration to coordinate the use of the medium. Routing in multi-hop MANETs is therefore different from routing in traditional IP networks. As described in Chapter 1 the routing protocols developed for MANETs can be divided into four broad categories: proactive, reactive, hybrid and hierarchical. Each has its own advantages and disadvantages. The proactive protocols attempt to keep up-to-date topological information of the entire network at each node through periodic broadcasts. When a node wants to send a packet to another node, a route is known and immediately available if any such route exists. Proactive protocols can consume a large portion of the available bandwidth to keep routing information up-to-date but the delay before sending a packet is minimal. The reactive protocols on the other hand request the route on demand when a packet needs to be sent. Most reactive protocols cache the acquired routes so that subsequent packet transfers do not generate the same route request in the near future. The cache entries become stale after a short period of time and are thus removed from the cache. Any subsequent request for the same route would result in a fresh acquisition of the route. Since the route acquisition can involve global flooding of the network, there can be considerable delay in sending the packet if the route information is not available in the cache. Reactive protocols tend to consume less bandwidth than proactive protocols as nodes do not acquire routing information for the routes that are not used. The hybrid approach is a compromise of the reactive and proactive approaches. Hybrid protocols provide greater flexibility to network users to tune their routing protocols according to their specific application needs. Depending on how much latency their application can withstand before its packets are sent on the network and the available bandwidth, users can tune their network to incorporate the appropriate amount of reactive and proactive elements in their routing protocol. In both reactive and proactive strategies, global flooding of the network cannot be

avoided. Since ad-hoc networks are bandwidth limited and their topology changes often, frequent global flooding can cause congestion in the network. Hierarchical routing protocols have been developed to reduce the cost of flooding the network. In these protocols not all nodes rebroadcast a packet, only some nodes called the *router* nodes are assigned the responsibility. These router nodes form a virtual backbone in the network and all traffic goes through some subset of router nodes. This chapter first details the characteristics of 802.11 networks and then describes some of the popular routing protocols.

2.1 802.11

As described previously, wireless networks have fundamental characteristics that make them significantly different from traditional wired networks. This requires the MAC (Medium Access Control) sublayer of wireless networks to incorporate functionality that is untraditional for MAC sublayers. This prompted the Institute of Electrical and Electronics Engineers, Inc. (IEEE) to develop a separate MAC standard for Wireless LANs. This standard is called 802.11 [Std99] and is based on IEEE 802.x LAN standards. 802.11 LANs can operate in two modes, infrastructure or managed mode and ad-hoc mode. In the infrastructure mode there exists a special node called a base station or Access Point which co-ordinates the access to the wireless medium, routes packets between wireless nodes and connects a wireless LAN to a wired network. All wireless stations wanting to be part of the wireless LAN have to associate with an Access Point. All messages exchanged within the wireless LAN or exchanged with other networks go through the Access Point. In the ad-hoc mode all stations are peers and there is no centralized administration to co-ordinate the access of wireless medium or routing. In this thesis we will concentrate on the ad-hoc operation of wireless LANs.

IEEE 802.11 [Std99] uses the same IEEE 802 48-bit address space and thus

IEEE 802.11 addresses are compatible with the address space used by the IEEE 802 LAN family. For IEEE 802.11, the length of the MAC service data unit (MSDU) must be less than or equal to 2304 bytes. When the length of the MSDU is greater than this value, *fragmentation* is accomplished at each immediate transmitter. Each immediate receiver performs *defragmentation* or recombines MAC Protocol Data Units (MPDUs) into a single MSDU. Frame sequence number and Fragment number fields in MPDUs help in defragmenting the right MSDU. Only MSDUs with a unicast receiver are fragmented. Broadcast/Multicast frames are not fragmented even if their length exceeds the threshold. The MPDUs resulting from the fragmentation of an MSDU are sent as independent transmissions, each of which is separately acknowledged. This permits transmission retries to occur per fragment rather than per MSDU.

Stations directly exchange control and data frames when operating in the ad-hoc mode. The fundamental access method used by stations in IEEE 802.11 networks is Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). When a wireless station has a packet to transmit, it senses the medium to determine if another station is transmitting. If the medium is not determined to be busy then the transmission may proceed. The CSMA/CA distribution algorithm mandates that a gap of a minimum specified duration exist between contiguous frame sequence, this is called Inter-Frame Space (IFS) and its value depends on the previous frame transmitted. IFS is important to prevent successive frames from interfering with each other. If the medium is determined to be busy, a station should defer until the end of the current transmission. After deferral (and prior to attempting to transmit again immediately after a successful transmission) the station selects a random backoff interval and decrements the backoff interval counter while the medium is idle. When transmitting a unicast message, a station exchanges short control frames (request to send (RTS) and clear to send (CTS)) with the receiver, after determining

that the medium has been idle for Distributed Inter Frame Space (DIFS) and after any deferrals or backoffs, prior to data transmission. The transmitter sends a request to send (RTS) control frame which is acknowledged by the receiver with a clear to send (CTS) control frame if it also determines that the medium is idle and is ready to receive data. This is followed by data transmission at the sender and reception at the receiver. The receiver then acknowledges that it has successfully received the data by sending an ACK control frame. Since no collision detection mechanism is available, ACKs are the only way the transmitter can find out that the receiver received the data correctly. RTS/CTS control frames are important to avoid the *hidden terminal* problem [FglA97]. Consider the wireless nodes in Figure 2.1. Nodes A and C do not know about each other's presence because they are not in range of each other. In absence of an RTS, CTS exchange they both could start data transmission simultaneously after sensing that the medium is idle. When the data reaches node B, it is garbled due to interference of the signals from both A and C. This is known as the hidden terminal problem as A and C are hidden from each other. When RTS/CTS exchange is employed before data transmission then C can find out that B would be receiving data when it hears the CTS frame sent by B in response to A's RTS frame and can defer its transmission. The RTS/CTS mechanism is not used for MPDUs with broadcast or multicast immediate addresses because there are multiple destinations for the RTS and thus potentially multiple concurrent senders of the CTS. The RTS/CTS exchange also performs both a type of fast collision inference and a transmission path check. If the return CTS is not detected by the station originating the RTS, the originating station may repeat this process more quickly than if the long data frame has been transmitted and a return ACK frame has not been detected. Note that there are no negative acknowledgements, so a receiver station may have received the frame correctly and the error might have occurred in the reception of the ACK frame. To the initiator of the frame exchange, this condition is indistinguishable from an error occurring in the initial frame.

Figure 2.1: Hidden Terminal Problem, Nodes A and C are hidden from each other

Each frame consists of the following basic components:

- i. A MAC *header*, which comprises frame control, duration, address and sequence control information;
- ii. A variable length *frame body*, which contains information specific to the *frame type*;
- iii. A frame *check sequence* (FCS), which contains an IEEE 32-bit cyclic redundancy code (CRC).

802.11 is further classified into 802.11a, 802.11b, 802.11d and 802.11g depending on the frequency band that the physical layer operates in and the data rate that is supported. 802.11b can support data rates of 5.5Mbps and 11Mbps in addition to the basic data rates of 1Mbps and 2Mbps. The physical layer of 802.11b operates in the 2.4GHz band. For the purpose of this work we assume 802.11b for the physical and data link layers.

2.2 AODV

Ad-Hoc On Demand Distance Vector (AODV) [PR99] is the currently most popular routing protocol for MANETs. In this protocol, a node discovers a route on demand, i.e., only when it is needed, and caches it. Network wide flooding is used to discover routes. The protocol requires that nodes maintain local connectivity information by sending periodic local (1-hop) broadcast messages known as *hello* messages. Through these *hello* messages a node becomes aware of its neighbours or nodes in its radio range. When a source node wants to send a message to a destination node and a route to the destination is not available in the cache, it initiates a path discovery process by broadcasting a route request (RREQ) packet. When a node receives a RREQ packet it checks whether it has received the same packet before, if it has then it discards the packet. The node then determines whether it has a route to the destination node in its cache. If it cannot satisfy the route request of the source then it rebroadcasts the packet after setting up a reverse path to the source. To set up a reverse path, a node records the address of the neighbour from which it received the first copy of RREQ. Eventually a RREQ will arrive at a node (possibly the destination itself) that possesses a current route to the destination. The node then unicasts a route reply (RREP) packet back to the source. As the RREP travels back to the source, each node along the path sets up a forward pointer to the node from which the RREP was received and updates its timeout information for the route entries to the source and destination. Nodes that are not along the path determined by the RREP will timeout after `ACTIVE_ROUTE_TIMEOUT` and will delete the reverse path to the source.

When a node detects that a node is unreachable (a link failure is detected either by failure to receive hello messages or a link-layer acknowledgement), the node upstream of the break propagates a route error (RERR) packet to all the active neighbours of the route for which the node was the next hop.

For each route entry a list of active neighbours is also maintained. A neighbour is considered active if it originates or relays at least one packet for that destination within the most recent `ACTIVE_TIMEOUT` period. All routes in the route table cache are tagged with destination sequence numbers which guarantees that no routing loops can form, even under extreme conditions of out-of-order packet delivery and high node mobility. The sequence number also helps in checking the freshness of a route, the greater the sequence number the more fresh a route is.

Several extensions have been proposed to the basic AODV routing protocol. Some of the most prominent ones have been accepted as part of standard AODV. One such modification is use of link layer feedback to maintain neighbourhood information instead of periodic *hello* messages. Another modification is use of expanding ring search for route request packets. Instead of sending a network wide broadcast for a RREQ, the source node starts out by sending a limited broadcast (done by setting the TTL (time to live) field in the packet to `TTL_START`). If this broadcast fails (indicated by a timeout) to find a route to the destination then the source increases the previous TTL value by `TTL_INCREMENT` and sends out another broadcast with the higher TTL value. This process is repeated till the TTL value reaches `TTL_THRESHOLD` after which the source sends out a broadcast with TTL equal to `NETWORK_DIAMETER`. If this broadcast also fails to discover a route to the destination then upto `RREQ_RETRIES` such broadcasts are sent again. If still a route cannot be found then all packets queued for that destination are dropped. When `RREQ_RETRIES` is 0, the timeout for each RREQ is calculated as:

$$Timeout = Min(2.0 * TTL * LINK_TRAVERSAL_TIME, MAX_RREQ_TIMEOUT)$$

Where `LINK_TRAVERSAL_TIME` is the time it takes to traverse a link and `MAX_RREQ_TIMEOUT` is the maximum possible value of timeout. When

RREQ_RETRIES is greater than 0 then the timeout of each RREQ is calculated as:

$$\text{Timeout} = \text{Min}(2.0 * \text{TTL} * \text{LINK_TRAVERSAL_TIME} * \text{RREQ_RETRIES}, \\ \text{MAX_RREQ_TIMEOUT})$$

2.3 DSR

Dynamic Source Routing (DSR) [JM96] is another reactive routing protocol and is similar to AODV in operation. The main difference between AODV and DSR is that DSR performs source routing, while AODV uses next-hop information stored in the nodes of the route. Source routing is a routing technique in which the sender of a packet determines the complete sequence of nodes through which to forward the packet; the sender explicitly lists this route in the packet's header, identifying each forwarding hop by the address of the next node to which to transmit the packet on its way to the destination node. The route discovery process in DSR is similar to AODV. When a node wants to send a packet to another host it checks its route cache for a route to the destination. If the route is not available in the cache then the node broadcasts a route request packet containing the identity of the destination. In addition to the address of the source and destination, each request packet contains a *route record*, in which is accumulated a record of the sequence of hops taken by the route request packet as it is propagated through the ad hoc network during route discovery. When the packet reaches a node that does not contain the route then it appends its address to the *route record* in the request packet and rebroadcasts the request. When the packet reaches a host (including the destination) that has a route to the destination, the host appends the route to the accumulated route record in the packet and sends a *route reply*. In order to return the *route reply* packet to the initiator of the route request packet, the node must have a route to the initiator. If it has a route entry for the initiator in its route cache then the *route reply* packet is unicasted to the initiator. Otherwise, the node can reverse the route in the route

record of the route request packet, and use this route to send the *route reply* packet. This, however, requires the wireless links to work equally well in both directions, i.e., the wireless links must be bidirectional. If this condition is not true, then the host can piggyback the route reply packet on a route request packet targeted at the initiator of the original route discovery.

2.4 DSDV

One of the first routing protocols for MANETs is Destination Sequenced Distance Vector (DSDV) [PB94], which can be called an adaption of the Bellman-Ford Distance Vector protocol for MANETs. Packets are transmitted between the stations of the network by using the routing tables which are stored at each node of the network. Each node's routing table lists all available destinations, and the number of hops to each. Each route table entry is tagged with a sequence number which is originated by the destination node. The sequence number serves the same purpose as in AODV, to avoid loops in the route and to indicate their freshness. To maintain the consistency of routing tables in a dynamically varying topology, each node periodically transmits updates in addition to transmitting updates when significant new information is available. Thus DSDV is a proactive protocol. Route advertisements are sent by broadcast or multicast. In order to reduce the amount of information carried by these advertisements, two types of packets are defined. One will carry all the available routing information, and is called a "full dump". The other type will carry only information changed since the last full dump, and is called the "incremental". Full dumps are transmitted infrequently when no movement of mobile hosts is occurring. When node movements become frequent and the size of an increment approaches the size of a network protocol data unit (NPDU), then a full dump can be scheduled. To further reduce the traffic the advertisements of the routes which may not have stabilized yet is delayed. When a mobile host receives new routing information, that information is compared to the information already

available from previous routing information packets. Any route with a more recent sequence number is used. Routes with older sequence numbers are discarded. A route with a sequence number equal to an existing route is chosen if it has a better metric such as smaller number of hops. When a link to the next hop of a route has broken, any route through that next hop is immediately assigned an ∞ metric and an updated sequence number. The modifications are immediately broadcasted in a routing information packet.

2.5 OLSR

Optimized Link State Routing (OLSR) [Rfc26] is based on the link state algorithm and comes under the proactive class of protocols. OLSR is an optimization over a pure link state protocol as it compacts the size of the information sent in the messages, and furthermore reduces the number of retransmissions to flood these messages in the entire network. For this purpose, the protocol uses a multipoint relaying technique to efficiently and economically flood its control messages. In a pure link state protocol, all the links with neighbour nodes are declared and are flooded in the entire network. The OLSR protocol is an optimization of a pure link state protocol in the sense that each node declares only a subset of nodes among its neighbours as its *multipoint relay sensors*. This reduces the size of control packets and also minimizes flooding of the control traffic as only the *multipoint relays* re-broadcast messages in the network. The idea is to minimize the flooding of broadcast packets in the network by reducing duplicate retransmissions in the same region. This technique significantly reduces the number of retransmissions in a flooding or broadcast procedure.

Apart from normal periodic control messages, the protocol does not generate extra control traffic in response to link failures and additions. The protocol keeps the routes for all the destinations in the network, hence it is beneficial for the traffic

patterns where a large subset of nodes are communicating with each other, and the [source, destination] pairs are also changing with time. The protocol is particularly suitable for large and dense networks, as the optimization done using the multipoint relays works well in this context. The protocol does not require a reliable transmission for its control messages: each node sends its control messages periodically, and can therefore sustain a loss of some packets from time to time.

Each node selects a set of neighbour nodes as its *multipoint relays* (*MPRs*). The neighbours of any node N which are not in its *MPR* set read and process each packet but do not retransmit the broadcast packet received from node N. Each node also maintains the set of its neighbours that have selected it as their *multipoint relay*. This set is called the *MPR Selectors* of the node. A node retransmits only those broadcast messages that come from its *MPR Selectors*. A node selects its multipoint relay set among its one hop neighbours in such a manner that the set covers (in terms of radio range) all the nodes that are two hops away and it has a bidirectional link to each node in the multipoint relay set. Each node in the network periodically broadcasts the information about its one-hop neighbours which have selected it as a multipoint relay. Upon receipt of this *MPR Selectors* information, each node calculates and updates its routes to each known destination. Therefore, the route is a sequence of hops through the multipoint relays from source to destination. A node maintains 2-hop neighbourhood information through *HELLO* messages which are sent periodically by each node. In the *HELLO* messages each node sends out its neighbour list. When a node receives neighbour lists from all of its neighbours then it has complete 2-hop neighbourhood information. Using this information a node computes its *MPR* set such that the nodes in the *MPR* set cover all the two hop neighbours and it has a 1-hop bidirectional link to all nodes in the *MPR* set. These selected *MPRs* are declared in the subsequent *HELLO* messages transmitted by a node, so that the information reaches the multipoint relays and they can con-

struct their *MPR Selectors* list. The multipoint relay set is recalculated whenever there is a change in 2-hop neighbourhood information.

All *MPR* nodes periodically broadcast *Topology Control (TC)* messages. These *TC* messages contain the *MPR Selector* list of a *MPR* node. These messages are rebroadcast only by *MPR* nodes. A node uses the information obtained from *TC* messages to construct its routing table. At a node *A*, the routing table is a next hop array indexed by the destination node. If *B* is an immediate (1-hop) neighbour of *A*, then $next-hop_A(B)$ is *B* itself. In general, if *A* receives a *TC* message that indicates that *D* is on the *MPR* selector list of *C* then $next-hop_A(D) = next-hop_A(C)$. *A* fills in its next-hop array using all *TC* messages it has received until no new destination can be added.

2.6 ZRP

The proactive protocols use excessive bandwidth to maintain routing information, while reactive protocols have long route request delays. In ad-hoc networks where the largest part of traffic is directed to nearby nodes, Zone Routing Protocol (ZRP) [HP97] can offer proactive performance to a zone centered on each node and reactive performance for nodes outside this zone. A routing zone is defined for each node separately, and the zones of neighbouring nodes overlap. The routing zone has a radius of ρ , expressed in hops. The zone thus includes all the nodes whose distance from the node in question is at most ρ hops. The locally proactive routing component is called the Intra-zone Routing Protocol (IARP), while the globally reactive routing component is named Inter-zone Routing Protocol (IERP). IARP and IERP are not specific routing protocols but any proactive routing protocol can be used as the IARP while any reactive protocol can be used as the IERP. The fact that the topology of the local zone of each node is known can be used to reduce traffic when global route discovery is needed. Instead of broadcasting packets, ZRP uses a concept called

bordercasting. Bordercasting utilizes the topology information provided by IARP to direct query requests to the border of the zone. A node that has a packet to send first checks whether the destination is within its local zone using information provided by IARP. In that case, the packet can be immediately routed. If the destination is outside the zone and its route is not known in the cache, then a *route request* packet is bordercast. If the receiver of a route request packet knows the destination, it responds by sending a route reply back to the source. Otherwise, it continues the process by bordercasting the packet, i.e., sending the packet to nodes that are at the border of its zone. Thus if the zone radius is one hop then we get a pure reactive protocol while if the zone radius approaches infinity then the protocol becomes a pure proactive one.

Chapter 3

Related Work

The problem of message delivery among disconnected Mobile Ad-hoc Networks is not new to the research community. In particular, the idea of using intermediate nodes to relay messages between disconnected hosts has been proposed earlier by Li and Rus [LR00]. Different research groups have approached the problem of delivery in a disconnected network from the perspective of different MANET applications and hence have come up with different solutions.

Li and Rus explored message delivery in disconnected MANETs where nodes can be instructed to move in order to transmit messages. Such an assumption may be valid for MANETs where nodes are robots deployed to perform sensing tasks in a remote or hazardous environment. If a node A wants to send a message to node B, who is out of range, A can move some distance till it comes in range of B and then send the message to B, or it can approach an intermediate node that can act as a messenger to send the message to B. For instance, A can move to approach C, C moves to approach D, then D moves and sends the message to B. Li and Rus present an algorithm to find the optimal trajectory for relaying a message in this fashion when the trajectories of all nodes are known. They also present a modified algorithm when the trajectories of all nodes are not known. According to

their algorithm a node A wanting to compute the optimal trajectory chooses the node B reachable in minimal time among the nodes which haven't been processed, and marks B as ready. Then A updates the current minimal time from A to all nodes that are not ready. The running time of the algorithm is $O(n^2)$, where n is the number of nodes. When the trajectories of the nodes are not known then their algorithm relies on a location update mechanism that a node uses to inform other nodes about its current position. This leads to two types of communication between nodes (including those that are not in range): one is true message communication, the other is location update. Both types of communication require that nodes change their trajectories. Although their work is interesting, the assumptions made about nodes changing trajectories and a priori knowledge of trajectories of all nodes are unrealistic and contradictory. Their algorithm does not deal with the situation where such active changes in trajectories leads to a deadlock like situation.

Karumanchi et. al. [KMP00] describe the problem of network partitioning in a MANET formed by a group of firefighters involved in a firefighting mission. Each firefighter is required to update its location information to servers in the network and must be able to obtain location information of other firefighters involved in the mission by querying the servers. Their solution employs quorum-based strategies to update and query information in a partitioned network. Their primary goal is to provide high availability of information from the distributed database in the face of network partitioning. Given a set S of servers, a quorum system is a set of m subsets of S , namely $S_0, S_1, \dots, S_{m-1}$ called quorums, such that the union of the m subsets is set S and the intersection of any two quorums is not null. The quorum is constructed a priori and every node knows the membership of these sets. When a node has to update its location information it randomly selects a quorum S_i from the set of quorums and sends the update message to all the servers in the quorum. When a node wishes to make a query it randomly selects a quorum S_j and sends a

query message to all the servers in the quorum. As two quorums always intersect, the set of queried servers is bound to contain at least one server that belonged to the quorum that received the latest update. Hence, each query returns the latest value of the queried data item. Although this is true if a node can communicate with every server, it may not be true for a partitioned network as the servers that were supposed to return the latest value of the queried data item may now be in some other partition of the network. Karumanchi et.al. provide heuristic solutions to handle this situation. Their heuristic solution consists of each node maintaining a list of unreachable servers. A node randomly selects a quorum and eliminates the servers that are not reachable or the node first eliminates all quorums that have at least one node in the unreachable list and randomly selects from the remaining quorums. They argue that using the former approach for queries and the later for updates provides the best results. Although their approach is suitable for certain applications it requires logic on both client and server side.

Wang and Li [WL02b] describe the problem of network partitioning among a group of mobile nodes that are requesting and downloading information on demand from a centralized service. Their goal is to provide service coverage even when the network is partitioned by replicating the service to appropriate nodes before a partition takes place. They employ a partition prediction scheme to predict partitioning in the network before it occurs [WL02a]. They describe a situation where their scheme could be useful. Consider a museum where visitors equipped with wireless devices gather to view an exhibition. The museum provides an electronic guide that contains information about items on exhibition and maps. The wireless devices held by the visitors collectively form an ad-hoc network. As the visitors move around to view the various items on display, they connect to the guide via multi-hop wireless links to request and download desired information on demand. However, as the visitors move around, the topology of the ad-hoc network changes dynamically.

Some visitors may lose access to the electronic guide when wireless links fail and the network partitions. They describe a group mobility model called Reference Velocity Group Mobility Model (RVGM) to model node movements in such situations. In this model each mobility group has a mean group velocity. Each member node's velocity is described as deviation from its mean group velocity. The node velocity distribution in each mobility group is modeled as a Gaussian distribution parameterized by the mean group velocity and a variance that represents the amount of variation that exists in the member node velocities. The nodes are able to compute their velocity using successive GPS (Global Positioning System) location information. A mobile node's group membership is dynamic, that is, it may switch mobility group at any time. A subset of the mobile nodes in the network host a critical service such as a web server or database repository. The clients query the service from the servers and piggyback their identifier, location, and velocity information on their service requests. To predict the partitioning in the network each server executes a sequential clustering algorithm to identify the different groups and their mean velocities. It then computes the relative velocity of each group with respect to its velocity and using this information a server can compute the time when a group will go out of range. A server replicates the service to the node most distant from its position in the direction of the relative velocity of the departing group. When the replication is due, the server first checks if the target node is reachable and is still a client and if both conditions hold, the server initiates the replication. The mobile clients run a distributed grouping algorithm at a regular interval to discover servers. If there are several, the client selects the best server through velocity comparison. Thus after a network partition the clients can easily discover the new server. Again their scheme requires logic at both client and server ends. Also their scheme can be applied only when node movements follow their mobility model.

Chapter 4

Algorithm

The *Carrier Select* algorithm [SH03]¹ is described from the points of view of each of the nodes that participate in it. The intuition behind this algorithm is that an undelivered message (destined to a disconnected node) should be disseminated farthest in every direction in the connected graph containing the source. Since in the scenarios we are considering nodes move at fairly slow speed, a node at the boundary of the connected subgraph has higher probability of coming in contact with another network partition than any other node in the subgraph. At the same time we would like to minimize the number of nodes used to reach the boundary to reduce the overhead of replication.

4.1 The Source Node

When a node S is unable to find a route to a node D or has lost the route to node D , it buffers the message m meant for D and requests the neighbour list of all of

¹The algorithm described in [SH03] is slightly different from the one described in this thesis. In the algorithm described in [SH03], a node always selects nodes that form an independent set as carrier nodes. It uses a greedy approach to select this independent set by first sorting the candidate nodes in descending order of the number of nodes they “see” that are outside the range of the selecting node. The selecting node picks the first node from the sorted list as carrier node and removes its neighbours from the list. And this process is repeated till the list is empty.

its neighbours. Based on the 2-hop neighbourhood information received from its neighbours, node S selects some of its neighbours as carrier nodes such that all of its 2-hop neighbours are covered. The algorithm at node S can be described as follows:

```

cand_carrier := nbr(S)
seen := nbr(S) ∪ {S}
carrier := ∅
while  $\max_{Z \in \textit{cand\_carrier}} |\textit{nbr}(Z) - \textit{seen}| > 0$  do
    Z :=  $\arg \max_{Z \in \textit{cand\_carrier}} |\textit{nbr}(Z) - \textit{seen}|$ 
    carrier := carrier ∪ {Z}
    seen := seen ∪ nbr(Z) ∪ {Z}
    cand_carrier := cand_carrier − {Z}
od
while  $\exists Z \in \textit{cand\_carrier}$  such that  $\textit{nbr}(Z) \cap \textit{carrier} = \emptyset$  do
    carrier := carrier ∪ {Z}
od
eliminated := cand_carrier − carrier

```

nbr(*S*) is the neighbour set of S not containing S. *cand_carrier* is the set of candidate carrier nodes, while *carrier* set is the set of neighbours selected for carrying the message

In the first while loop node S selects those nodes as carrier nodes that “see” (are in radio range of) at least one node that has not been “seen” before. A greedy approach is followed in selecting these nodes starting from the node that “sees” most nodes outside the range of S. The rationale behind it is that selecting a node that is connected to more 2-hop neighbours increases the chances of propagating the message to the boundary of the partition.

In the second while loop node S selects the remaining nodes that are not neighbours of each other and are not in radio range of any of the carrier nodes selected in the first while loop. This is done to select boundary nodes when node S is one hop away from the boundary. For example, consider Figure 2.1, suppose B has an undelivered message and would like to disseminate it in its connected graph. In Figure 2.1 its connected graph contains only two nodes A and C. If B only uses the first while loop to select carrier nodes then its *carrier* set would be $\emptyset$ because none of its neighbours “see” any node that is not in range of B. The second while loop selects an independent set (nodes that are not in range of each other) among the remaining neighbours of node B. In this example, the independent set is {A,C}. Selecting an independent set guarantees that nodes are far apart from each other and at least one of them is far apart from B to justify the replication of message on them.

After selecting the carrier nodes, node S sends a HOLD control message to each node in the *carrier* set.

```
<message-type, message-sequence-number, source,
                                destination, m, nbr(S)>
```

The *message-type* field indicates that this is a HOLD message.

The nodes in the *eliminated* set are sent a NACK for message *m* which they buffer in their NAKMSG queue. Node S and the nodes in the *carrier* set buffer message *m* in their HOLDMSG queue. Thus the message is replicated not only on the boundary nodes of the connected subgraph containing the source but also on all the intermediate nodes used to reach the boundary. This extra redundancy is provided because it is possible that the direction of movement of a boundary node is not the same as the position of the node as seen from the source.

4.2 Other Selected Nodes

When a node Y receives a HOLD control message from another node X, it starts a similar selection process to select other nodes to hold the application message m . Like the original source node, node Y considers its neighbours as potential carrier nodes. However, some nodes are known to have already participated in the selection process for this message. Thus, node Y can eliminate from consideration the node it received the message from (X), the original source node (S), and those nodes that are in the $nbr(X)$ set sent by X in the HOLD control message. Node Y requests the neighbour list from the remaining potential carrier nodes.

A neighbour list request message, NBREQ, consists of the following fields:

`<message-type, message-sequence-number, source, destination>`

A node T receiving a NBREQ message from node Y replies with a neighbour list reply message, NBREP. A NBREP control message sent by a node T consists of the following fields:

`<message-type, message-sequence-number, source, destination,
nbr(T), status>`

The status field in the NBREP message reports on the status of this message at the node that sent the reply, and can be one of three values: NEW meaning that this node knows nothing of the message, NACKED, meaning that this node has previously been sent a NACK for the message, or HELD, meaning that this node is holding the message.

Nodes responding with NACKED or HELD in their status are eliminated from further consideration by Y. After node Y receives the neighbour list from the nodes, it uses the same algorithm (see Section 4.1) as the source node S to select other nodes to hold message m . The only difference is in the initialization of the sets *cand_carrier* and *seen*. The set *cand_carrier* contains only those neighbouring nodes

of Y that have not been eliminated from consideration by Y in any of the above steps. If $held(Y)$ is the set of neighbours of Y that responded with status HELD and $nacked(Y)$ is the set of neighbours of Y that responded with status NACKED then the algorithm at node Y can be described as follows:

```

cand_carrier := nbr(Y) - nbr(X) - {X} - {S} - held(Y) - nacked(Y)
seen := nbr(Y) ∪ nbr(X) ∪ {X} ∪ {Y} ∪ {S} ∪ held(Y) ∪ nacked(Y)
carrier := ∅
while  $\max_{Z \in cand\_carrier} |nbr(Z) - seen| > 0$  do
     $Z := \arg \max_{Z \in cand\_carrier} |nbr(Z) - seen|$ 
    carrier := carrier ∪ {Z}
    seen := seen ∪ nbr(Z) ∪ {Z}
    cand_carrier := cand_carrier - {Z}
od
while  $\exists Z \in cand\_carrier$  such that  $nbr(Z) \cap carrier = \emptyset$  do
    carrier := carrier ∪ {Z}
od
eliminated := cand_carrier - carrier

```

Again the nodes in the *carrier* set are sent a HOLD message containing the message m and its neighbour set, $nbr(Y)$. The nodes in the *eliminated* set are sent a NACK for the message.

In order to bound the size of the HOLD control message, the $nbr(Y)$ set sent by node Y contains only those nodes that are its neighbours. Therefore, the upper bound on the size of the $nbr(Y)$ set is the maximum number of neighbours a node can have. The alternative of accumulating neighbour sets as the HOLD message propagates would eliminate some redundancy, at the expense of having HOLD messages that could be as large as the number of nodes in any partition.

Figure 4.1a shows an example network partition of 10 nodes. Suppose node

Figure 4.1: (a) A network partition of 10 nodes (b) Node positions after some mobility

0 wants to send a message to a node outside the network partition. It initiates the process of selecting carrier nodes by first requesting the neighbour list from its neighbouring nodes 1, 2, 3, 4, and 5. After execution of the *Carrier Select* algorithm its *carrier* set consists of nodes $\{2, 4\}$ while its *eliminated* set consists of $\{1, 3, 5\}$. When node 2 receives the HOLD message it starts the selection process by eliminating nodes 0, 1 and 3 from consideration and requesting the neighbour list from nodes 6 and 7. After execution of the algorithm its *carrier* set consists of node $\{6\}$. When node 6 receives a HOLD message it starts the selection process but ends up eliminating all nodes from its neighbour list and thus the process terminates at this node. Nodes 4 and 9 are the other nodes that hold the message.

After a while, if node 0 has another application message for an unreachable host, the nodes that are selected this time may be different from the ones previously selected as some of the nodes have moved to different positions as shown in Figure 4.1b. When node 0 executes the algorithm it again ends up with the *carrier*

set {2, 4} but this time the eliminated set does not contain node 3. This time the eliminated set is {1, 5}. When node 2 starts its selection process, it requests the neighbour list from nodes 3 and 6. Node 3 is also being considered by node 4 at the same time. If node 3 receives a HOLD or a NACK message for the same application message from node 4 before the NBREQ message from node 2 arrives at node 3 then node 3 reports its status appropriately to node 2 either stating that it already holds the message (status=HELD) or it has been eliminated from consideration (status=NACKED) by some node (in the present scenario node 3 would receive a HOLD message from node 4). In that case node 2 will eliminate node 3 from its candidate node set. However, if HOLD message from node 4 arrives after node 3 has responded to NBREQ message from node 2 then node 3 will remain in the candidate node set of node 2. Ideally we would like a node to receive either a NACK or a HOLD control message for a particular application message and not both. This cannot be achieved because of distributed nature of the algorithm and the changing neighbour lists of the nodes during the execution of the algorithm. Thus it is possible for a node to receive both a NACK and a HOLD message for the same application message and in any order. In such a situation, the HOLD message always overrides the negative acknowledgement.

4.3 Message Delivery

Once every FIND_DST_INTERVAL, each node tries to find a route to the destinations of the messages it has stored in its HOLDMSG queue. If it is able to find a route to a destination then it delivers all the messages stored for the destination in its HOLDMSG queue. When a destination node receives a message, either from the original source or from an intermediate node, it sends an acknowledgement to the sending node. For each message stored in the HOLDMSG queue of a node A, the id of the node B that sent the HOLD message to node A and the nodes that were sent a HOLD message by node A are also kept. When an acknowledgement is received

for a message in the HOLDMSG queue the node forwards the acknowledgement to all such nodes. These forwarded acknowledgements help in removing messages that have been successfully delivered to the destination from the HOLDMSG queues of other nodes. This potentially prevents a lot of route search packets (in case of reactive routing protocols) being broadcast by the nodes holding the message. Some of the nodes to which the acknowledgements are forwarded may be disconnected from the forwarder node. In that case the acknowledgement is either dropped or a selection process similar to the one used for the data messages is started depending on how important the acknowledgements are in a particular application. A message is removed from a HOLDMSG queue if an acknowledgement for the message is received or the message has been in the HOLDMSG queue for longer than MAX_HOLD_TIME.

4.4 Neighbour List Request Timeout

After a node sends a NBREQ message it is possible that the neighbour that was supposed to receive the request message has wandered away or is down. To handle such cases a node re-requests a neighbour for its neighbour list, each time waiting for NB_LIST_REP_TIME seconds after sending the neighbour list request message, NBREQ. If a neighbouring node does not respond with a NBREP message even after making MAX_NBREQ_ATMPTS requests for its neighbour list, then the node proceeds with the selection process considering that the non-responsive node is no longer its neighbour.

Chapter 5

Design

The schemes described in Chapter 3 are complicated and not general enough to be applicable in all scenarios. In addition, some of the schemes make assumptions that are unrealistic. Our goal has been to come up with a protocol that is simple and general enough to be applicable in all scenarios and does not make any additional assumptions other than those that are inherent in any MANET.

Instead of designing a new routing protocol to handle the disconnected case we decided to use the existing routing protocols and develop a simple strategy that can be used with any of the existing routing protocols including those described in Chapter 2. The routing protocols has to make an upcall to transfer an undelivered message to our protocol. Our protocol takes the undelivered message and employs a distributed algorithm to replicate the message on selected nodes. As with all systems ours is also based on some key assumptions. The assumptions are fundamental to the operation of any MANET and are explicitly or implicitly made by all MANETs.

1. A Node has knowledge of the IP address and the MAC address of the disconnected destination. This could have been obtained from the destination either offline or from a previous direct message transfer (through 0 or more intermediate nodes).

2. Hosts do not continuously move so rapidly as to make the flooding of every packet the only possible routing protocol. In particular we assume that nodes move at fairly slow speed compared to the time it takes to replicate a message across the network.
3. All hosts wishing to communicate with other hosts within the ad hoc network are willing to participate fully in the protocols of the network.
4. All hosts are using the same channel for communication.

We do not assume that nodes know their own position, the position of other nodes, or the complete topology of the network.

Although we implemented our system on a discrete-time event-driven simulator (see Chapter 6) we describe the multi-threaded design of our system here. Figure 5.1 depicts the high level design of our system. Figure 5.2 shows the position of the Voilà layer in the protocol stack. There can be other implementations of our system such as an extension of TCP but we haven't explored that in this thesis. We use UDP as the transport layer protocol in the current prototype.

During normal operation, the Voilà layer adds a 104 bit header to a transport layer segment before giving it to the routing layer and rips off the same header at the receiving end before giving it to the transport layer. The header that is added is depicted in Figure 5.3. The header consists of a 8 bit type field indicating the type of the message. All 0s in the type field indicate a normal application packet. Voilà identifies each application packet with a unique 32 bit sequence number. The source and destination addresses are the IP addresses of the source and destination of the message. Multicast and Broadcast packets are left untouched by Voilà.

When a message cannot be delivered to the destination, the routing protocol

makes an upcall and hands over the message to the upcall thread of Voilà. The upcall thread acquires the neighbourhood information from the underlying routing protocol, fills a temporary data structure called the *pending queue*, sends the NBREQ message to the neighbours and starts the NB_REQ_EXP timer. If the NB_REQ_EXP timer has expired MAX_NBREQ_ATMPTS times (see Section 4.4) or Voilà thread has received NBREP messages from all the neighbours, it uses *Selection Elimination logic* to select carrier nodes for the undelivered message and sends the HOLD messages. The *Selection Elimination logic* contains the *Carrier Select* algorithm or any other algorithm (see Section 6.3) to select the carrier nodes. When a HOLD message is received for an application message, it is put in the HOLDMSG queue and the *Selection Elimination logic* is used again to select the carrier nodes. Expiry of FIND_DST timer (see Section 4.3) starts the message delivery process for the items in the HOLDMSG queue and expiry of MAX_HOLD timer removes old messages from the HOLDMSG queue.

Figure 5.1: High level architecture of Voilà

Figure 5.2: System architecture with Voilà layer

Figure 5.3: Header added by Voilà layer

Chapter 6

Evaluation and Results

6.1 Routing Protocol Parameters

We implemented our protocol on the sns [SNS03] simulator which is a staged implementation [WgS03] of the ns2 [ISINS] simulator. We use a modified Random Way Point Model, described in the next section of this thesis, to model the node movements in our experiments. The range of each wireless node in the simulations is 100m. The traffic generator consists of an application running at each node that tries to send a message of 64 bytes once every five seconds to a random node (other than itself). The underlying routing protocol used is AODV. Table 6.1 shows the values of various parameters of AODV used in the simulations.

Expanding ring search (see Section 2.2) is used to discover a lost or an unknown route and the EXPANDING_RING_SEARCH option in the configuration file is used to regulate it. The options TTL_START, TTL_THRESHOLD, NETWORK_DIAMETER and RREQ_RETRIES have the same meaning as described in Section 2.2. AODV_LOCAL_REPAIR option is used to turn on/off the local repair of routes. In the experiments, *Hello* messages are used instead of the passive link layer detection mechanism to maintain neighbourhood information at nodes. This is because the protocol would like a full neighbourhood information rather than information about only active neighbours. The *Hello* messages are sent every second

by a node and a node considers a neighbour out of range if no message is received from the neighbour for 3 consecutive hello intervals. The HELLO_INTERVAL and ALLOWED_HELLO_LOSS options regulate these parameters.

6.2 Mobility Model

The Random Way Point model is the most widely used mobility model for testing protocols in mobile ad-hoc networks. We have made two modifications to the Random Way Point model to make it more practical.

6.2.1 Random Way Point Model

The Random Way Point Model was first described in [JM96]. Since then it has been widely used to test MANET protocols in simulated environments. In this model nodes move around in a “room”. Each node starts from an initial position selected randomly from within the simulated area. As the simulation progresses, each node pauses at its current position for a constant period of time and then randomly chooses a destination location from within the simulated region and moves there by selecting a random speed uniformly from the interval $(0, V]$, where V is the maximum speed with which a node can move.

Table 6.1: AODV parameter values used in the simulations

Parameter	Value
EXPANDING_RING_SEARCH	ON
TTL_START	7
TTL_THRESHOLD	7
NETWORK_DIAMETER	30
RREQ_RETRIES	1
AODV_LOCAL_REPAIR	OFF
AODV_LINK_LAYER_DETECTION	OFF
HELLO_INTERVAL	1
ALLOWED_HELLO_LOSS	3

6.2.2 Modifications in Random Way Point Model

We believe that the applications developed for Mobile Ad-hoc environments will be mostly interactive and therefore people using them would be walking or strolling. Studies [RM80] have shown that the normal walking speed of an adult human being is around 1.5 m/s. We also consider the possibility of vehicles (particularly in the disaster relief scenarios outlined in Chapter 1) moving at a slow speed that can participate in the communications. Yoon, Liu and Noble [YLN03] show that selecting a minimum speed greater than zero for mobile nodes is necessary to attain a meaningful steady state. Therefore we choose node speed from a normal distribution with mean speed set to 1.5 m/s. The minimum speed is set to 0.4 m/s while the maximum speed is set to 5.0 m/s.

The Random Way Point Model assumes uniform distribution of nodes, which is quite “unrealistic”. The density of nodes would vary a lot in any realistic scenario including those described in Chapter 1. There would always be some hot-spots where nodes are clustered. In a disaster relief scenario these hot-spots could be sites where relief and rescue work is being carried out; on a beach these could be particularly scenic points, snack bars, or volleyball courts; in a military zone these could be various army camps. In our model we have five such hot-spots randomly distributed in a space of 3000m x 600m. Each hot-spot is a circular disc of radius 250m. Each node is initially placed at a random position in a randomly selected hot-spot. Nodes then, instead of selecting a destination uniformly from the whole area, select a random position within a randomly selected hot-spot. There is also a small probability (10%) of choosing a destination which is not in a hot spot.

6.3 Results

To evaluate the performance of our protocol we compare it against other protocols that can be used to aggressively deliver messages to disconnected destinations. These protocols are described in the next subsections.

6.3.1 Source-only Protocol

In the Source-only protocol, the source node buffers the messages that could not be delivered to a disconnected host. It tries to find a route to the destinations of the buffered messages once every `FIND_DST_INTERVAL`. Those messages that cannot be delivered within `MAX_HOLD_TIME` are removed from the buffer. Thus the Source-only protocol is the limiting case of our protocol where there is only one carrier node for each undelivered message, the source itself.

6.3.2 Oracle Protocol

In the Oracle protocol every node is omniscient, i.e., it has the global knowledge of the trajectories of all the nodes in the network and hence can choose the optimal intermediate node as the carrier node, if any such node exists. Based on the trajectories of the nodes, the Oracle protocol selects a node from the connected subgraph containing the source that will come in contact with the destination within the `MAX_HOLD_TIME`. If such a node exists then the undelivered message is sent to the node otherwise it is dropped. Clearly the assumption of knowledge of the trajectories of all nodes in the network is unrealistic and is impossible to achieve in practice. Therefore, no practical protocol can match the performance of Oracle protocol in terms of number of messages successfully delivered to the destination. That's why we refer the Oracle protocol as the optimal protocol. It is useful to compare to the Oracle protocol to see how much improvement is possible for a protocol.

6.3.3 Voilà-GPS

When nodes are equipped with GPS receivers, they can obtain their geographical position information. Periodic position information can help a node in computing its speed and direction of movement. This information can allow a node to make better decisions about carrier nodes for an undelivered message. From this point, we will refer to the algorithm described in Chapter 4 as Voilà-basic and an algorithm that takes advantage of the velocity of a node to select carrier nodes as Voilà-gps.

Consider Figure 6.1, if node 0 is running Voilà-basic then it will select all six of its neighbours as carrier nodes. Such a selection could turn out to be a bad decision, for instance when all nodes are heading towards the same destination. In such a situation it is highly likely that they will come in contact with the same nodes and so replicating the message on one of them should have been enough. As is obvious, node velocity would have been a better metric for making the replication decision in this case. While it was not possible to obtain the velocity information in Voilà-basic, it is possible to obtain that information in Voilà-gps.

Voilà-gps algorithm at node X can be described by the following steps.

1. Node X sends a broadcast containing the list of neighbours that need not respond with their velocity information, i.e., those neighbours that have already been eliminated as potential carrier nodes for the message.
2. The neighbours that are not in the list and those that have not received a NACK for the same undelivered message before, unicast their velocity information back to the source.
3. Node X puts all nodes that respond with their velocity information in a set of potential carrier nodes. Node X then selects a neighbour Z moving with the maximum speed as the carrier node from the set of candidate

carrier nodes. It then eliminates all nodes from its set of potential carrier nodes that are moving in almost the same direction as Z, i.e., those nodes whose velocity vector makes an angle less than `DEGREE_CONST` with the velocity vector of Z. This process is repeated until the set of potential carrier nodes is empty. Figure 6.2 shows the angle between the velocity vectors of two nodes.

4. The source node then sends a NACK to the eliminated nodes and a HOLD message to the selected nodes.

We use broadcast for item 1 in the above list because it eliminates the requirement of maintaining a full neighbour set at the node. Now a node can maintain a partial neighbour list and can still make all decisions of selecting the carrier nodes. Thus, if the nodes are using AODV as the routing protocol, they can turn on `LINK_LAYER_DETECTION` and not use periodic *Hello* messages for maintaining the neighbourhood information at a node (see section 6.1). The rationale behind choosing the node with maximum speed as the carrier node from the set of potential carrier nodes is that the node with the greatest speed can go farthest and/or reach a target in the least amount of time.

6.3.4 Random Selection

We also compare our scheme against a Random Selection scheme. In the Random Selection scheme a node selects some percentage of its neighbours as carrier nodes at random. Each neighbour has equal probability of being selected as a carrier node. Each selected node is sent a HOLD message as in all other schemes. But the rejected nodes are not sent any NACK as its not necessary. The percentage of neighbouring nodes that are chosen as carriers is determined by the parameter `RANDOM_CONST` in the config file. In the Random Selection protocol a carrier node sends the set of nodes selected as carriers as the eliminated set in the HOLD

Figure 6.1: All six neighbours selected as carrier nodes in Voilà-basic

Figure 6.2: Angle between the velocity vectors of nodes 1 and 2 is 90 degrees

Figure 6.3: Selection of Carrier nodes when RANDOM_CONST is 50%

message. This is done so that a node can eliminate these nodes from consideration. Such nodes are considered to be part of the node's selection set and are included in the percentage that must be satisfied by a node. Unlike the other schemes, the eliminated set from previous hops is added to the selected set at a node before being put in the HOLD message. This is done to help a node better identify previously selected nodes. However, it's not possible to guarantee that a node would have only RANDOM_CONST percentage of neighbouring nodes as carrier nodes for a message because of the distributed nature of the protocol and sharing of neighbours between nodes. For example in Figure 6.3 RANDOM_CONST is 50%, therefore node 0 chooses nodes 1 and 2 as carrier nodes, node 1 chooses node 4 as carrier node while node 2 chooses node 3 as carrier node. Node 3 is also a neighbour of node 1, so node 1 ends up with all its neighbours as carrier nodes. Therefore RANDOM_CONST can be considered only as a lower bound on the number of neighbours that can be selected as carrier nodes.

6.4 Discussion

Figure 6.5 shows the performance of the five protocols when the density of nodes is varied (Figure 6.6 shows the same graph without the Oracle protocol). The values of the parameters listed in Table 6.2 are kept constant for all experiments described in this section. The performance is measured in terms of the number of “carried” messages received by a destination node. Here “carried” messages are those messages that are buffered by a “carrier” node (either by the source itself or by some other node) in its HOLDMSG queue (see Section 4.1 and Section 4.3) in any of our active delivery protocols and are subsequently delivered to the destination within MAX_HOLD_TIME (see Section 4.3). In the absense of an active delivery mechanism these packets would have been dropped by the routing protocol. From Figure 6.5 it is clear that Oracle can achieve much more performance gain than other protocols as node density increases (Figures 6.6 6.9 6.12 have been included to show the differences of the other protocols’ performances more clearly). We were expecting Voilà-gps to perform much better than Voilà-basic but it performs only slightly better than Voilà-basic and the reason for this is the nature of our geographical space. The geographical space (see Figure 6.4) is quite narrow (600m wide) and the cluster (see Section 6.2.2) diameter is large (500m), further we require nodes to select destinations within a cluster 90% of the time. While this is a believable scenario for a beach, it limits the value of a velocity vector. For instance, the velocity vector of a node cannot have a pure y component unless it chooses a destination within its cluster or very close to its cluster. Figure 6.4 shows a snapshot of the positions of the nodes in the simulated region. The five circles indicate the five clusters and the rectangular boundary encloses the simulated geographical area (3000m x 600m). Although no protocols are able to keep up with Oracle as the density of nodes increases, Voilà-basic and Voilà-gps are able to achieve 50% of Oracle’s performance in low density situations. The values of some of the other parameters used in this experiment are listed in Table 6.3. Figure 6.7 shows the

percentage of messages that cannot be delivered to the destination when no aggressive delivery scheme is used and Figure 6.10 shows the percentage of undelivered messages carried by each aggressive delivery protocol to their respective destinations.

Figure 6.8 compares the performance of the five protocols as `FIND_DST_INTERVAL` (see Section 4.3) is varied (Figure 6.9 shows the same graph without the Oracle protocol). In this experiment the number of nodes simulated is 70 while `MAX_HOLD_TIME` is calculated by the formula $(2 * \text{FIND_DST_INTERVAL} + 5)$. This formula is used to allow a carrier node to make two attempts to find the destination of a buffered message before it is discarded from its `HOLDMSG` queue. As `FIND_DST_INTERVAL` is increased, we see a rise and fall in the performance of all protocols except Oracle. This can be explained by the network wide broadcast messages that are sent out every time a route to the destination of a buffered message is requested. When `FIND_DST_INTERVAL` is small the frequency with which these routes are requested is high, which causes congestion in the network and a lot of collisions in the shared air space. As `FIND_DST_INTERVAL` is increased the number of these broadcast messages decreases, consequently congestion and collisions decrease and performance improves. But as the `FIND_DST_INTERVAL` increases beyond a certain point, some of the messages remain undelivered as the route is not requested in time. The Oracle protocol, being aware of trajectories of all nodes, can instantly deliver a message when the destination joins its connected subgraph.

Figure 6.11 compares the performance of the five protocols as `MAX_HOLD_TIME` is varied (Figure 6.12 shows the same graph without the Oracle protocol). Again, in this experiment the number of nodes is 70. The `FIND_DST_INTERVAL` is kept constant at 20 seconds for this experiment. As expected, with increase in `MAX_HOLD_TIME`, all protocols can deliver more “carried” messages to the destinations. Again we see the performance of the Oracle increases at a much faster

rate than the other protocols. As the MAX_HOLD_TIME increases, the number of routes requested per FIND_DST_INTERVAL also increases and so there is an increase in the broadcast traffic in the network which causes more collisions and thus tends to limit the rate of performance gain.

A question may arise as to why the performance decreases as the FIND_DST_INTERVAL is decreased from 40 seconds to 20 seconds while performance increases when MAX_HOLD_TIME is increased. While the broadcast messages increase in both cases, the rate at which they increase differs. When FIND_DST_INTERVAL is decreased from 40 seconds to 20 seconds the rise in broadcast messages is almost linear. But when MAX_HOLD_TIME is increased linearly the rise in broadcast messages is less than linear because there are only a constant number of possible destinations. The number of disconnected destinations also does not change drastically as the nodes move at fairly slow speed. The only time that drastic change occurs when a node becomes disconnected from its connected subgraph or joins another connected subgraph. The number of nodes that can undergo such changes within 125 seconds is pretty low therefore we see an increase in the performance despite an increase in the broadcast messages in the network.

Figure 6.13 shows the change in performance of Voilà-gps as DEGREE_CONST (see Section 6.3.3) is varied. As explained previously the nature of the simulated geographic region limits the value of the velocity vectors of the nodes and therefore changing the DEGREE_CONST has little effect on the performance of Voilà-gps:

Figure 6.14 shows the change in performance of Random Selection protocol as the value of RANDOM_CONST is varied. As the value of RANDOM_CONST increases the number of carrier nodes for each message increases and so the probability that the message would be delivered to the destination within MAX_HO-

Figure 6.4: A typical snapshot of positions of nodes in the simulated region of 3000m x 600m

Table 6.2: Parameter values kept constant throughout all experiments

Parameter	Value
Simulated region	3000m x 600m
Number of clusters	5
Radius of each cluster	250m
Transmission range of each node	100m
Simulation Time	1 hour
NB_LIST_REP_TIME	1.1 seconds
MAX_NBREQ_ATMPTS	2

LD.TIME also increases and therefore we see rise in the performance of Random Selection protocol. For these experiments the number of nodes is 70 and the values of FIND_DST_INTERVAL and MAX_HOLD_TIME are 20 seconds and 45 seconds respectively.

Table 6.3: Parameter values used in the density experiment

Parameter	Value
FIND_DST_INTERVAL	60 seconds
MAX_HOLD_TIME	125 seconds
RANDOM_CONST	50 %
DEGREE_CONST	20 degrees

Figure 6.5: Number of messages delivered as density of nodes is varied

Figure 6.6: Number of messages delivered as density of nodes is varied (without Oracle protocol)

Figure 6.7: Percentage of undelivered messages when no aggressive delivery mechanism is used

Figure 6.8: Number of messages delivered as FIND_DST_INTERVAL is varied

Figure 6.9: Number of messages delivered as FIND_DST_INTERVAL is varied (without Oracle protocol)

Figure 6.10: Percentage of undelivered messages carried by each protocol to the destinations

Figure 6.11: Number of messages delivered as MAX_HOLD_TIME is varied

Figure 6.12: Number of messages delivered as MAX_HOLD_TIME is varied (without Oracle protocol)

Figure 6.13: Number of messages delivered as DEGREE_CONST is varied

Figure 6.14: Number of messages delivered as RANDOM_CONST is varied

Chapter 7

Future Work

Although we have been able to achieve reasonable performance gain with our scheme, we believe its only a fraction of what can be achieved. To achieve more performance gain it's imperative to develop a better understanding of the movement of the nodes. Insights into the mobile behaviour of the nodes can help to instill more intelligence into the replication strategy. Such intelligence would help to select the "right" carrier node(s) that will come in contact with the destination in the near future without requiring each node to possess a priori knowledge of the trajectories of all the nodes. Although several attempts have been made to model the mobile behaviour of nodes ([CBD02] [JM96] [HGPC99] [WL02a] [YLN03] [JRAS03]) in an ad-hoc environment, the models are mostly simplistic and of only theoretical importance. A user study that traces the movement of mobile nodes would certainly help to develop such insights. We are not aware of any such user studies being carried out at this time. Although such a user study would present several financial and technological challenges, it would help in developing the insights necessary to model a "realistic" mobile behaviour of nodes in an ad-hoc setting. Such a model would not only help a replication scheme like ours but also improve the routing protocols that have been mostly tested using simple mathematical mobility models.

Although we have tried to explore the parameter space as much as possible, we have not explored the interdependence of the parameters. Also some of the parameters have been kept constant like the routing protocol. From the results shown in Section 6.4 it is clear that network wide broadcast is the major cause of collision and congestion in the network. AODV does not have any mechanism to minimize this effect. There are other routing protocols such as OLSR (see Section 2.5) and more recently MOCDS [AWF02] that use relay nodes to check the broadcast storm. It would be interesting to obtain performance results when one of these protocols is used as the underlying routing protocol.

Like most protocols developed for MANETs we have also analyzed the performance of our protocol on a simulator. Using the simulator allows us to quickly examine the correctness, scalability and the effects of various parameters on the protocol. But a simulator also makes some simplifying assumptions about the environment. It would be interesting to test our protocol on a real testbed and understand other “real world” factors that might affect our protocol.

Chapter 8

Conclusion

The most important contribution of this thesis is that it has demonstrated how an aggressive message delivery protocol can improve the delivery of messages in MANETs. It also presents an intelligent scheme that can be used for aggressive delivery of undelivered messages within any existing ad-hoc routing protocol. Our scheme is unique in the manner in which it captures “reality”. We have made very realistic assumptions about the capabilities of a mobile node, its mobile behaviour and any a priori information it has about other nodes. We are able to demonstrate that for a sparse network, the performance of our scheme is close to 50% of the optimal scheme. Apart from this, the thesis also demonstrates the weaknesses in the popular mobility models and the routing protocols being used for MANETs. The mobility models make simplistic assumptions and are not able to capture reality. We have made modifications to a popular mobility model to make it more realistic but further research and/or a user study is needed to capture realistic mobile behaviour of wireless nodes. The widely used routing protocols for 802.11 networks create a broadcast storm when requesting routes for a destination. Recently, researchers have proposed alternatives to remove this drawback but at this time none of them have been analyzed enough to be used as the basis for further research.

Bibliography

- [AWF02] K. Alzoubi, P. Wan and O. Frieder, "Message Optimal Connected Dominating Set Construction for Routing in Mobile Ad Hoc Networks", 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02), 2002.
- [BMJHJ98] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols". In *Mobile Computing and Networking*, 1998, 85-97.
- [CBD02] T. Camp, J. Boleng, and V. Davies, "Mobility models for ad hoc network simulations", in *Wireless Communication and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2002.
- [FglA97] C. L. Fullmer and J. J. Garcia-Luna-Aceves, "Solutions to Hidden Terminal Problems in Wireless Networks", *Proceedings of ACM SIGCOMM'97*, Sep 1997, 39-49.
- [HGPC99] X. Hong, M. Gerla, G. Pei, and C. Chiang, "A Group Mobility Model for Ad Hoc Wireless Networks," in *Proceedings of the 2nd ACM International Workshop on Modeling and Simulation of Wireless and Mobile Systems*, 1999.

- [HP97] Z. J. Haas and M. R. Pearlman, "The Zone Routing Protocol(ZRP) for Ad hoc Networks", Internet draft - Mobile Ad hoc Networking (MANET) Working Group of the Internet Engineering Task Force (IETF), Nov 1997.
- [ISINS] <http://www.isi.edu/nsnam/ns/>
- [JM96] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," Mobile Computing, Kluwer Academic Publishers, 1996.
- [JRAS03] A. Jardosh, E. Royer, K. Almeroth, S. Suri, "Towards Realistic Mobility Models For Mobile Ad hoc Networks", Proceedings of the Ninth ACM/IEEE International conference on Mobile Computing and Networking (Mobicom 2003), San Diego, USA, Sept 2003.
- [KMP00] G. Karumanchi, S. Muralidharan and R. Prakash, "Information Dissemination in Partitionable Mobile Ad Hoc Networks", in Proceedings of IEEE Symposium on Reliable Distributed Systems, Lausanne, Switzerland, Oct 2000.
- [LR00] Q. Li and D. Rus, "Sending Messages to Mobile Users in Disconnected Ad-hoc Wireless Networks," in Proceedings of the Sixth ACM/IEEE International conference on Mobile Computing and Networking (Mobicom 2000), Aug 2000, 44-55.
- [PB94] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," in SIGCOMM'94, 1994.
- [PHO02] D. Perkins, H. Hughes, and C. Owen, "Factors affecting the performance of ad hoc networks", in Proceedings of the IEEE International Conference on Communications (ICC), 2002.

- [PR99] C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing," in 2nd IEEE Workshop. Mobile Comp. Sys. and Apps., , Feb 1999, 90-100.
- [PRDM01] C. E. Perkins, E. M. Royer, S. R. Das and M. K. Marina, "Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks", IEEE Personal Communications Magazine, special issue on Mobile Ad Hoc Networks, Vol. 8, No. 1, Feb 2001, 16-29.
- [RM80] P. S. Rodman and H. M. McHenry, "Bioenergetics and the origin of hominid bipedalism", Americal Journal of Physical Anthropology, Vol. 52, 1980, 103-106.
- [RT99] E. M. Royer and C-K. Toh, "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks", IEEE Personal Communications Magazine, April 1999, 46-55.
- [Rfc26] RFC 3626, <http://hipercom.inria.fr/olsr/rfc3626.txt>.
- [SH03] R. Shah, N. C. Hutchinson, "Delivering Messages in Disconnected Mobile Ad Hoc Networks", 2nd International Conference on Ad Hoc Networks and Wireless, Springer LNCS 2865, Oct 2003, 72-83.
- [SNS03] <http://www.cs.cornell.edu/people/egs/sns/>
- [Std99] ANSI/IEEE Std 802.11, 1999 Edition.
- [THBSR02] J. Tian, J. Hähner, C. Becker, I. Stepanov, K. Rothermel, "Graph-based Mobility Model for Mobile Ad Hoc Network Simulation", Proceedings of 35th Annual Simulation Symposium, San Diego, USA, April 2002.
- [WL02a] K. Wang and B. Li, "Group Mobility and Partition Prediction in Wireless Ad-hoc Networks", in Proceedings of IEEE International

Conference on Communications (ICC 2002), Vol. 2, April-May 2002, 1017-1021.

- [WL02b] K. Wang and B. Li, "Efficient and Guaranteed Service Coverage in Partitionable Mobile Ad-Hoc Networks." in Proceedings of IEEE INFOCOM 2002, Vol. 2, Jun 2002, 1089-1098.
- [WgS03] K. Walsh and E. Gun Sirer, "Staged Simulation for Improving the Scale and Performance of Wireless Network Simulations", Proceedings of the Winter Simulation Conference, USA, Dec 2003.
- [YLN03] J. Yoon, M. Liu, and B. D. Noble, "Random waypoint considered harmful", in Proceedings of INFOCOM '03, April 2003.