2016

Routing and Security in Mobile Ad Hoc Networks

Baban A. Mahmood
University of Kentucky, baban@netlab.uky.edu
Author ORCID Identifier:
http://orcid.org/0000-0002-3787-265X

Digital Object Identifier: https://doi.org/10.13023/ETD.2016.492

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student’s advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student’s thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Baban A. Mahmood, Student

Dr. Dakshnamoorthy Manivannan, Major Professor

Dr. Miroslaw Truszczynski, Director of Graduate Studies

Routing and Security in Mobile Ad Hoc Networks

ABSTRACT OF DISSERTATION

A dissertation submitted in partial
fulfillment of the requirements for
the degree of Doctor of Philosophy
in the College of Engineering at the
University of Kentucky

By
Baban Ahmed Mahmood
Lexington, Kentucky

Director: Dr. Dakshnamoorthy Manivannan, Associate Professor of Computer Science
Lexington, Kentucky 2016

ABSTRACT OF DISSERTATION

Routing and Security in Mobile Ad Hoc Networks

A Mobile Ad hoc Network (MANET) consists of a set of nodes which can form a network among themselves. MANETs have applications in areas such as military, disaster rescue operations, monitoring animal habitats, etc. where establishing fixed communication infrastructure is not feasible. Routing protocols designed for MANETs can be broadly classified as position-based (geographic), topology-based and hybrid. Geographic routing uses location information of nodes to route messages. Topology-based routing uses network state information for route discovery and maintenance. Hybrid routing protocols use features in both position-based and topology-based approaches. Position-based routing protocols route packets towards the destination using greedy forwarding (i.e., an intermediate node forwards packets to a neighbor that is closer to the destination than itself). If a node has no neighbor that is closer to the destination than itself, greedy forwarding fails. In this case, we say there is void. Different position-based routing protocols use different methods for dealing with voids. Topology-based routing protocols can be classified into on-demand (reactive) routing protocols and proactive routing protocols. Generally, on-demand routing protocols establish routes when needed by flooding route requests throughout the entire network, which is not a scalable approach. Reactive routing protocols try to maintain routes between every pair of nodes by periodically exchanging messages with each other which is not a scalable approach also. This thesis addresses some of these issues and makes the following contribution.

First, we present a position-based routing protocol called Greedy Routing Protocol with Backtracking (GRB) which uses a simple backtracking technique to route around voids, unlike existing position-based routing protocols which construct planarized graph of the local network to route around voids. We compare the performance of our protocol with the well known Greedy Perimeter Stateless Routing (GPSR) protocol and the Ad-Hoc On-demand Distance Vector (AODV) routing protocol as well as the Dynamic Source Routing (DSR) protocol. Performance evaluation shows that our protocol has less control overhead than those of DSR, AODV, and GPSR. Performance evaluation also shows that our protocol has a higher packet-delivery ratio, lower end-to-end delay, and less hop count, on average, compared to AODV, DSR and GPSR. We then present an on-demand routing protocol called "Hybrid On-demand Greedy Routing Protocol with Backtracking for Mobile Ad-Hoc Networks" which uses greedy approach for route discovery. This prevents flooding route requests, unlike the existing on-demand routing protocols. This approach also helps in finding routes that have lower hop counts than AODV and DSR. Our performance evaluation confirms that our protocol performs better than AODV and DSR, on average, with respect to hop count, packet-delivery ratio and control overhead.

In MANETs, all nodes need to cooperate to establish routes. Establishing secure and valid routes in the presence of adversaries is a challenge in MANETs. Some of the well-known source routing protocols presented in the literature (e.g., Ariadne and endairA) which claim to establish secure routes are susceptible to hidden channel attacks. We address this issue and present a secure routing protocol called SAriadne, based on sanitizable signatures. We show that our protocol detects and prevents hidden channel attacks.

KEYWORDS: MANET, Position-Based Routing, On-demand Routing, Hybrid Routing, Secure Routing.

Author's signature: Baban Ahmed Mahmood

Date: December 9, 2016

Routing and Security in Mobile Ad Hoc Networks

By
Baban Ahmed Mahmood

Director of Dissertation: Dr. Dakshnamoorthy Manivannan

Director of Graduate Studies: Dr. Miroslaw Truszczynski

Date: December 9, 2016

Dedicated to the soul of my mother and father.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Dakshnamoorthy Manivannan, for his constructive comments, guidance, support, counseling, and advice throughout my graduate study. Thanks to Dr. Jun Zhang, Dr. Zongming Fei, and Dr. Avinash Sathaye for being members on my dissertation committee, and providing critical and constructive comments on my dissertation.

I would like to thank all my family members and my wife, Lanja, for their endless love and great patience during my PhD study at University of Kentucky.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

A mobile ad hoc network (MANET) consists of a set of nodes each of which is capable of being both a client and a router. The nodes form a network among themselves without the use of any fixed infrastructure, and communicate with each other by cooperatively forwarding packets on behalf of others. Routing protocols designed for MANETs need to be scalable, secure, robust, and have low routing overhead. Routing protocols designed for MANETs can be broadly classified as geographic routing protocols and topology-based routing protocols. Figure 1.1 shows a brief classification of routing protocols for MANETs [5].

1.1 Position Based Routing Protocols

In geographic routing protocols, nodes do not maintain information related to network topology (i.e., they are topology independent). They only depend on the location information of nodes to forward packets. Generally [6], nodes need their own location, their neighbors' locations, and the location of the destination node to which the packet needs to be forwarded. Using this location information, routing is accomplished by forwarding packets hop-by-hop until the destination node is reached [7]. Greedy forwarding, like the one used in GPSR [8], is one of the main strategies used in geographic routing protocols. In Greedy forwarding, an intermediate node on the route forwards packets to the next neighbor node that is closer to the destination than itself.

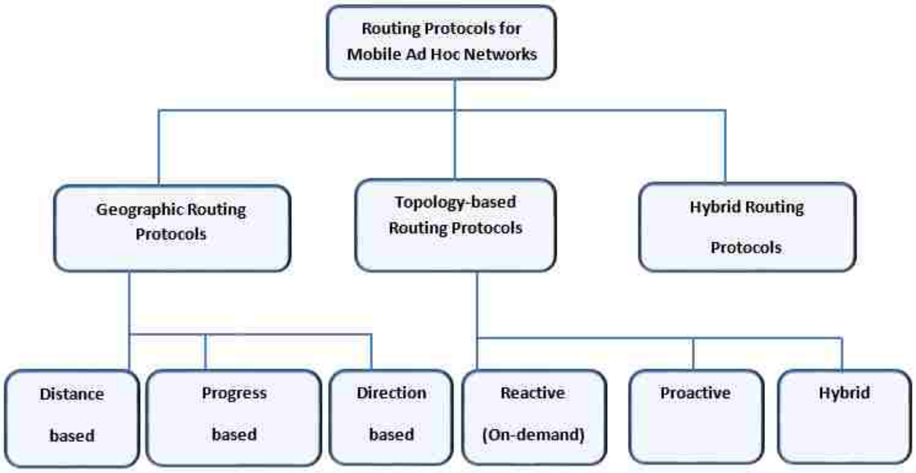

Routing Protocols for Mobile Ad Hoc Networks
Geographic Routing Protocols
Topology-based Routing Protocols
Hybrid Routing Protocols
Distance based
Progress based
Direction based
Reactive (On-demand)
Proactive
Hybrid

Figure 1.1: Routing Protocols for MANETs.

Different geographic protocols which use different greedy forwarding strategies are described below.

- Distance Strategy. Euclidean distance is mainly used in greedy forwarding to minimize the distance packets traverse from the source node to the destination node.

- Progress Strategy. This strategy tries to maximize the distance between the source node S and the projection A' of a neighboring node A (chosen for forwarding) onto the straight line joining S and the destination node D as shown in Figure 1.2. The larger the distance between a node holding a packet and the neighboring node's (chosen for forwarding) projection, the faster will be the progress of the packet towards the destination (S forwards to node A in Figure 1.2). The Most Forward within Radius (MFR) strategy used in [9] is an example of routing protocol using this strategy. Another protocol that uses the progress strategy is the Nearest with Forward Progress (NFP) [10] protocol.

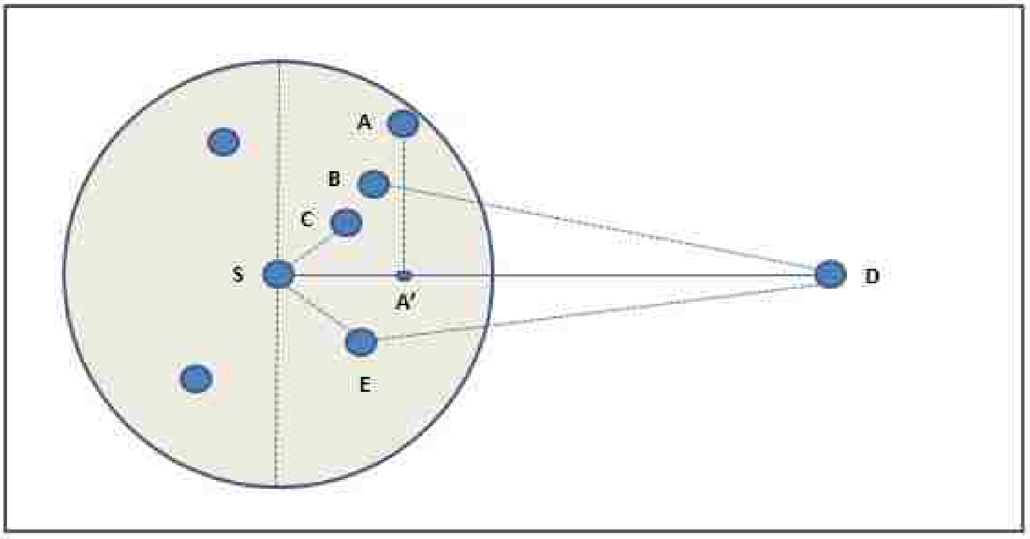


Figure 1.2: Greedy Routing Strategies [1]

.

Under NFP, a sender S forwards packets to the nearest neighboring node that is closer to the destination node than S (e.g., S forwards to node C in Figure 1.2).

- Direction-based Strategy. This strategy is also called compass routing [11]. It minimizes the spatial distance that packets travel by using deviation as its criteria to forward packets. The deviation is the angle between the line connecting the source node and the next hop and the straight line connecting the source node and the destination node. Under this approach, the neighbor that is closer to the line joining the source node and the destination node is selected as next hop.

1.2 Topology Based Routing Protocols

Topology based routing protocols depend on current topology of the network. Topology-based routing is also known as table-based routing. Topology-based routing can be classified into proactive routing protocols, reactive routing (on-demand) protocols, and hybrid routing protocols [12].

In Proactive Routing Protocols, like DSDV [13], nodes use pre-established table-based routes [14]. Therefore, routes are deemed reliable and nodes do not wait for route discovery which cuts off latency. However, overhead incurred for route construction and maintenance can severely degrade performance, limit scalability, and the routing table may consume lot of memory as the network grows. In addition to these limitations, frequent topology changes due to node movement may lead to out-dated or stale routes in the routing table which may result in packet loss.

In Reactive Routing Protocols, also called on-demand routing protocols, senders find and maintain a route to a destination only when they need it. Thus, reactive routing protocols such as AODV [15] try to establish a route to a destination only when needed. Reactive routing needs less memory and storage capacity than proactive protocols. However, in network areas where nodes can move more unpredictably and frequently, route discovery may fail since the route can be long and links may break due to node mobility or when facing obstacles [16]. Moreover, the delay caused by route discovery for each data traffic can increase latency.

On the other hand, Geographic Routing Protocols require only the location information of nodes for routing. They do not require a source node to establish a route to the destination before transmitting packets. Unlike on-demand routing protocols, they do not depend on flooding route request messages to establish routes. This feature helps geographic routing protocols reduce the extra overhead imposed by topology constraints for route discovery [2, 7]. A node only needs to know the position of its neighbors and the position of the destination to forward packets. This helps the protocols adapt to any topology changes and link failures easily since the next hop is decided locally. Therefore, geographic routing protocols generally are more scalable than topology based routing protocols [17, 18, 1].

1.3 Hybrid Routing Protocols

Hybrid routing protocols [19] combine features of both position-based routing and topology-based routing protocols. These features complement each other such that the resulted hybrid routing protocol is loop free and scalable, incurs lower control overhead, and can find a path to the destination as long as the network is not partitioned.

1.4 Secure Routing Protocols

A secure routing protocol enables nodes to exchange control information and data in the presence of adversaries which try to disrupt the functioning of the routing protocol. Several mechanisms have been proposed to provide secure routing for MANETs ([20] presents a survey of secure routing protocols).

Generally, attacks on routing protocols in MANETs fall into one of the following two categories [20]:

1. Resource-consumption attacks. In this category, an attacker injects packets into the network attempting to consume network and/or node resources such as bandwidth, memory, and computation power.

Injecting extra data packets into the network is an example of resource consumption attacks. When forwarded, these packets consume bandwidth unnecessarily. Another example is when an attacker *injects extra control packets* into the network. When nodes process and forward these control packets, more bandwidth and/or computational resources are consumed than those consumed by injecting extra data packets.

2. Routing-disruption attacks. In this category, an attacker tries to route legitimate data packets in dysfunctional ways.

1.4.1 Attacks on Routing in MANETs

To develop a good and secure routing protocol, one needs to understand the possible type of attacks on routing protocols. Below, we explain some attacks designed to disrupt routing protocols in MANETs.

- ***Routing Loop***

  An attacker sends forged routing packets causing data packets to traverse nodes in a cyclical path without reaching their destinations. This attack consumes energy and bandwidth in addition to causing data packets loss [21].

- ***Blackhole Attack***

  In this attack, a malicious node responds to a route request packet claiming that it has a valid and fresh route to the destination node. In this case, an attacker could trick a sender into routing all data packets to the attacker which discards them.

- ***Grayhole Attack***

  It is a special case of a blackhole attack where an attacker could create a gray hole where it selectively drops some control and/or data packets. For example, forwarding some data packets and all control packets or forwarding all control packets but not data packets [21].

- ***Blackmail Attack***

  In MANETs, nodes can keep track of perceived malicious nodes in a *blacklist* at each node, similar to watchdog and pathrater [22]. An attacker could blackmail (report) a good node, telling other nodes to add that *legitimate node* to their blacklists. This attack results in isolating legitimate nodes from the network.

- ***Gratuitous Detour Attack***

  An attacker may also attempt to cause a node to use detours (suboptimal routes) or may attempt to partition the network by injecting forged routing packets to prevent one set of nodes from reaching another. An attacker may attempt to make a route through itself appear longer by adding virtual nodes to the route [21].

1.5 Applications of MANETs

MANETs have applications in areas such as military, disaster rescue operations, monitoring animal habitats, etc. where establishing fixed communication infrastructure is not feasible or the preexisting infrastructure has been destroyed by a disaster or in

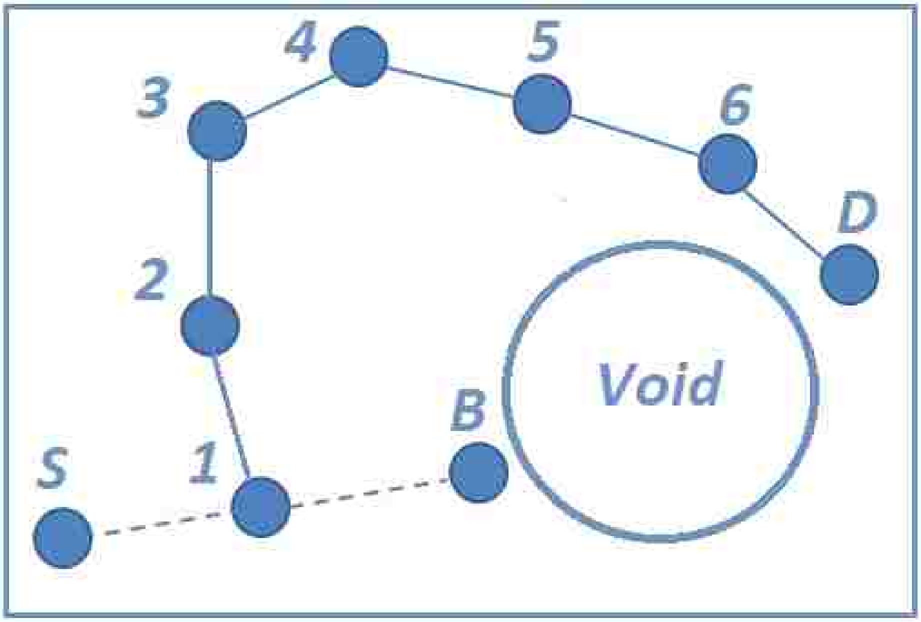


Figure 1.3: Example of Dead End in Greedy Forwarding.

war [16, 23, 24, 25, 26, 27]. Compared to wired networks, establishing MANETs is less expensive and hence are attractive.

1.6 Problems Addressed and Solved in this Dissertation

Greedy forwarding, the primary packet forwarding strategy used by geographic routing protocols, may fail in low density networks, networks with non-uniformly distributed nodes, and/or networks where obstacles are present. Therefore, the main problem with greedy forwarding strategy is that it does not guarantee packet delivery to the destination because of the dead end phenomenon even if there is a route to the destination. Figure 1.3 shows an example of dead end (void) problem. When the source node S needs to send packets to the destination node D, it forwards the packets greedily to a node that is closer to the destination than itself. On receiving the packets, each subsequent node does the same. When the packets reach node B,

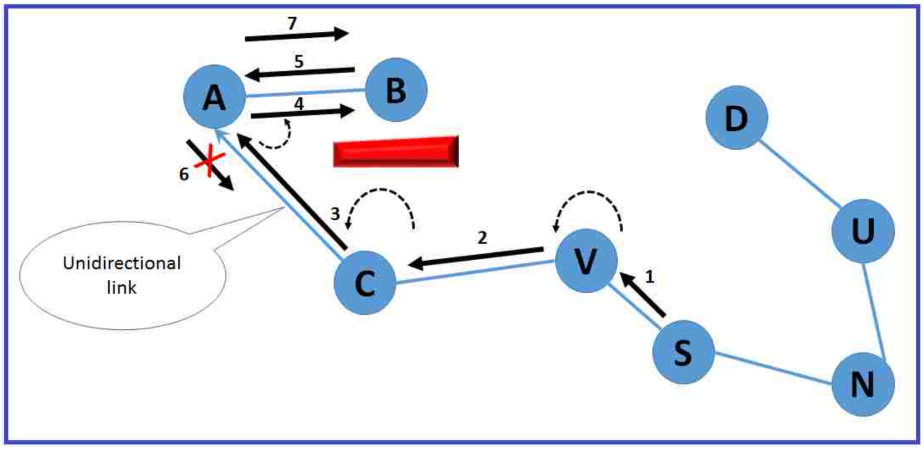


Figure 1.4: Example of Planarization where a Unidirectional Link Causes Routing Failure.

it finds that none of its neighbors are closer to the destination node D than itself and D is outside the transmission range of node B. This means there is a void between B and D in the direction towards D as shown in Figure 1.3. Even though there is a valid path from S to D through intermediate nodes *1,2,3,4,5,* and *6*, greedy forwarding cannot use it. This means, under pure greedy forwarding, packets may be dropped even though there are valid paths to destination nodes.

Generally, position-based routing protocols use planarization and face routing [8, 18] to go around voids. Planarization involves constructing the planar graph of local network. Graphs are generally planarized using the Gabreil Graph (GG) [28] or the Relative Neighborhood Graph (RNG) [29]. However, planarization may fail to generate bidirectional, connected, and/or cross link free local graphs as observed by Kim et al. and Frey et al. [30, 31]. This may be the result of node's incorrect estimate of its location or irregular communication range as a result of radio-opaque obstacles or transceiver differences.

As shown in Figure 1.4, a unidirectional link can cause an infinite loop during face

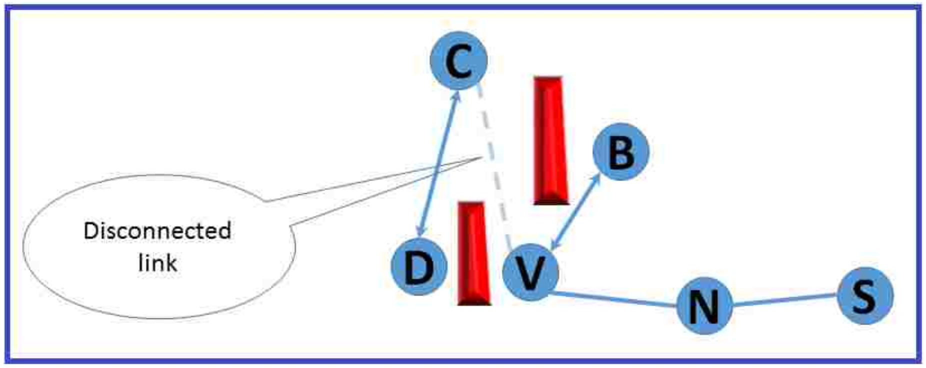


Figure 1.5: Example of Planarization where a Disconnected Link Causes Routing Failure.

traversal. In this example, when the source node S needs to send data packets to the destination node D, based on greedy forwarding, it sends that data packet to node V. Node V is a dead end for that packet, hence it switches to face routing and forwards the data packets to node C. In this case, when C constructs the planar graph of the local network using GG, it cannot see the witness B in the circle whose diameter is the distance between A and C because of the obstacle shown in Figure 1.4. Therefore, C generates a link to node A. However, node A does not create a link to node C in its local graph because it can see the witness B in the circle. Therefore, node C can forward the data packets to node A which in turn forwards them to node B and based on face routing, node B returns the packet to node A. Since node A does not have a link to node C in the local graph, it returns the packets to node B and as a result, the data packets loop.

Routing can also fail because of disconnected links as shown in Figure 1.5. In this example, source S needs to send data packets to destination D. When packets arrive at node V, they face a dead end and switch to face routing. From node V's view, B is a witness and from C's view, D is a witness. Therefore the link between V and C

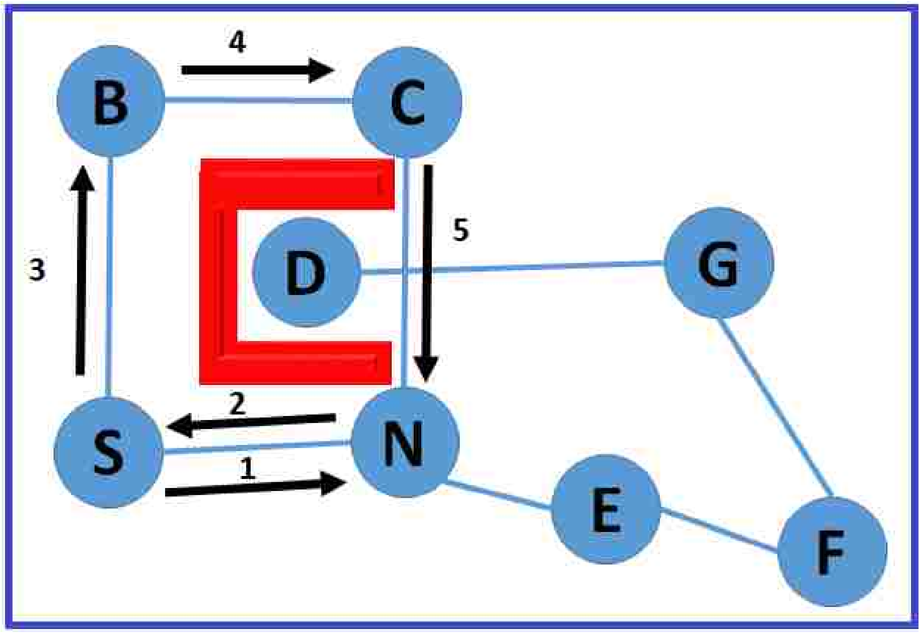


Figure 1.6: Example of Planarization where a Cross Link Causes Routing Failure.

is removed in the planarized graph of local network and as a result, the local graph is disconnected and packets cannot travel to D.

Another scenario where routing may fail is shown in Figure 1.6. In this example, when node S needs to send data packets to destination D, it first forwards the packet to next closer neighbor N which faces void and hence, creates a local planner graph. The obstacle shown in Figure 1.6 hides D from both N and C and as a result, a link between these two nodes is created which crosses the link between D and G. Then, based on face routing and right hand rule, N returns the packet back to S which in turn forwards it to B. From B, the packet is forwarded to C and then to N where it loops.

This means that face routing cannot always forward packets when they face void even when alternative valid paths exist. We address this problem and present an algorithm which uses a simple backtracking technique to route around voids.

MANETs are vulnerable to attacks that aim to disrupt the functionality of the routing protocols. The main problem occurs when adversarial nodes intentionally hide intermediate nodes to create routes that are not valid. Several secure routing protocols have been proposed in the literature to detect this hidden channel attack. However, many of them do not detect and prevent all types of attacks. We address this issue and propose a novel, secure routing protocol for MANETs. Following are the contributions of the dissertation.

1. ***GRB*** When pure greedy forwarding is used, packets are dropped when they face voids making recovery procedures necessary. The existing protocols mainly depend on face routing [8, 18] to deal with voids. However, in addition to failing to go around voids in some scenarios as explained above, face routing can incur high processing cost and high end-to-end delay. We address these issues and propose Greedy Routing Protocol with Backtracking for MANETs (GRB). GRB [32] is a novel and simple position based routing protocol which allows each node to forward data packets to its best neighbor possible until the destination is reached. Unlike GPSR, GRB uses less computation to determine the next hop when the packet faces a void.

2. ***HGRB*** Many of the on-demand routing protocols, such as DSR [33] and AODV [15], flood route requests throughout the network for route discovery which results in high control overhead due to redundant propagation of route requests. On the other hand, geographic routing protocols construct a planarized graph of the local network to route around voids using that graph,

resulting in high control overhead. This dissertation addresses these issues and proposes a novel on-demand routing protocol called Hybrid Greedy On-demand Routing Protocol with Backtracking (HGRB). HGRB [34] inherits the best of both topology-based and position-based routing paradigms. HGRB uses geographic approach for forwarding route requests (RREQs) during route discovery and uses simple backtracking to forward RREQs around voids.

3. ***SAriadne*** The existing secure source routing protocols such as Ariadne [21] and endairA [3] are prone to hidden channel attacks. Adversarial nodes can shorten the actual route by hiding genuine nodes from the route which results in creating invalid routes that are accepted by source nodes as valid routes. This dissertation addresses this issue and presents a novel protocol based on sanitizable signatures called SAriadne [35] that establishes secure and valid routes in MANETs.

1.7 Organization of the Dissertation

The rest of this dissertation is organized as follows: In Chapter 2, the related work and their merits and demerits are presented. In Chapter 3, Greedy Routing Protocol with Backtracking for MANETs (GRB) [32], and its performance evaluation results are presented. In Chapter 4, Hybrid On-demand Greedy Routing Protocol with Backtracking for MANETs (HGRB) [34], and its performance analysis are presented. In Chapter 5, SAriadne [35], A Secure Source Routing Protocol to Prevent Hidden-Channel Attacks, and its security analysis are presented. Finally, summary and conclusion are presented in Chapter 6.

Chapter 2 Related Work

In this chapter, we discuss recent works related to the work presented in the dissertation.

In the past, several researchers have surveyed geographic and hybrid routing protocols. Cadger et al.'s survey [17] discusses different design issues in geographic routing as addressed by different routing protocols [7, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45]; they also focus on how the existing geographic routing protocols address issues such as security [46, 47, 48], mobility of nodes [49, 50, 51], power consumption [52, 53], and quality of service [54, 55]. Manghsoudlou et al. [1] discuss different greedy forwarding strategies and recovery mechanisms used in geographic routing protocols. They focused on discussing different strategies that can be used as alternatives to face routing. The greedy forwarding strategies discussed by them are [9, 10, 56]; they also survey protocols [8, 38, 37, 57, 58, 59] using different variations of face routing using traversals along planar graphs; other variations discussed include geometric recovery strategy [60], flooding based handling of voids [61, 62], cost-based category [63, 64, 65], heuristic void handling [66, 67], and hybrid strategy [68]. Mauve et al. [69] discuss two major areas: geographic routing protocols and location service protocols. They compare the protocols in each area with other protocols in the same area. Different forwarding strategies discussed include greedy forwarding protocols [9, 10, 56], restricted directional flooding [70], hierarchical routing [67, 71], and grid routing protocols [66]. They surveyed different location service protocols [14, 72, 68, 73, 74]

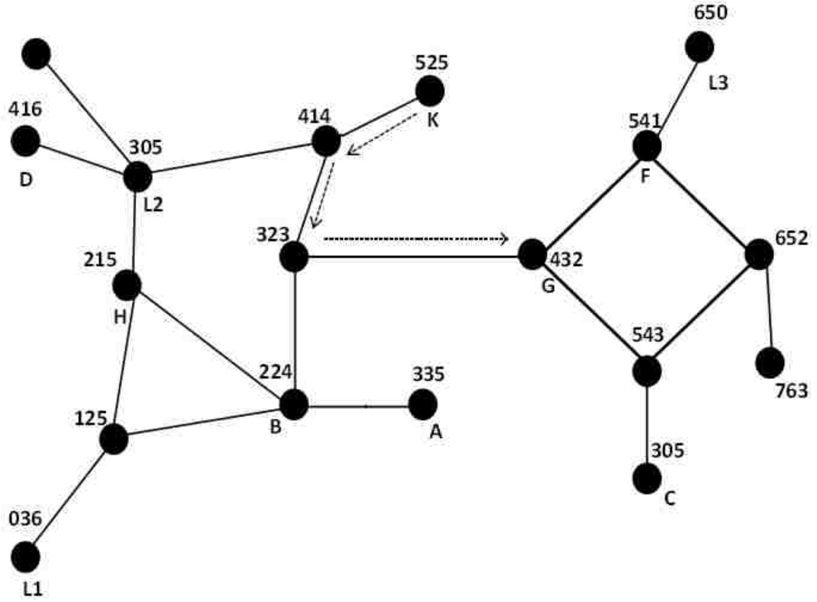


Figure 2.1: Example Illustrating a Node's Hop ID (a Vector) [2].

also.

In the rest of the dissertation, we interchangeably use the terms "dead end", "void", "local maximum", "local minimum", and "obstacle". All these terms mean that there is no node closer to the destination than the current node and the destination node is outside the transmission range of the current node.

2.1 Position-Based Routing Protocols

In this section, we review several geographic routing protocols that use nodes' position information for routing data packets.

2.1.1 GPSR: Greedy Perimeter Stateless Routing for Wireless Networks

GPSR [8], a well known geographic routing protocol proposed by Karp and Kung, uses greedy forwarding as the default forwarding strategy. When a packet confronts a void (i.e., when greedy forwarding fails), they planarize the local topology graph either by constructing the Relative Neighborhood Graph (RNG) or Gabriel Graph (GG) of the graph and use those graphs to route around the void. However, constructing the graph could be time consuming and results in high control overhead. Also, when GPSR faces a void it may fail to go around the void when the dead end node has no neighbors other than the one that sent the packet.

We address these issues and propose and novel greedy forwarding with backtracking technique that does not need to construct graphs and can forward packets around voids.

2.1.2 Hop ID: A Virtual Coordinate-Based Routing for Sparse MANETs

Zhao et al. [2], proposed a routing protocol called Hop ID Routing (HIR). In this protocol, the authors try to solve the dead end problem using virtual coordinate-based routing. They build a multidimensional coordinate system based on which they find the Hop ID distance between each pair of nodes in the network. The protocol selects specific nodes in the network, as landmark nodes. Each node has a hope ID, a vector of length equal to the number of preselected landmarks. The entry in each dimension of the node's vector represents the distance from that node to one of the landmarks. Figure 2.1 has three landmarks namely $L1$, $L2$, and $L3$. Since there are three landmarks, each node will be associated with a three dimensional vector. Node H, for example, is 2 hops away from $L1$, 1 hop away from $L2$, and 5 hops away from $L3$; therefore, its vector is (215). A predefined hash function is used to map each node to one of the landmarks. The authors used a distance function, called power distance, which helps to eliminate many dead ends. In HIR, each node can get a unique ID by hashing its IP address. Two different techniques are proposed to select Landmarks: Random Landmark Selection and Peripheral Landmark Selection.

In random landmark selection, a specific node C which is relatively stationary is chosen to become a coordinator. The coordinator C then uses a hash function to generate m random IDs, called landmark IDs. Node C then floods a CENTER packet (which contains the landmark IDs) throughout the network. Upon receiving the CENTER packet, each node inserts its upstream node ID and rebroadcasts the CENTER packet. Using this flooding method, a tree rooted at the coordinator C is built.

Peripheral Landmark Selection is similar to the Random Landmark Selection except that only the perimeter nodes are allowed to send CANDIDATE packets. A node becomes perimeter node if it is farther away from the coordinator C than its neighbors. In this technique, a shortest path tree rooted at C is built by finding the perimeter nodes.

This protocol does not scale well because it uses a central packet flooding technique initiated by a designated node to select LANDMARKs. Selecting landmarks results in proportionally high control overhead. However, data packet forwarding process does not produce additional control overhead to the protocol.

2.1.3 A Novel Location-Fault-Tolerant Geographic Routing Scheme for Wireless Ad Hoc Networks

Lin and Kus [6] proposed a location-fault-tolerant geographic routing protocol (LGR) using both traditional geographic routing and position-based clustering technique. Mobile nodes are assumed to obtain location information using some location service. However, since nodes may move at any time and location update may not be triggered in time, inaccurate location information can be used, causing routing problems. Wrong greedy decision and planarization collapse are two intrinsic problems. Wrong decision incurs routing loops and packet dropping. In a planar graph, there are no cross links among the nodes preventing routing loops. Planarization collapse, on the other hand, degrades performance and may result in routing loops. All these problems occur due to incorrect location information. To overcome such deficiencies, LGR benefits from fixed cluster positions instead of node's accurate reported locations.

The network area is divided into several polygons, each called a cluster. Each cluster has a center with coordinates defined as cluster position that could be used as cluster ID. A node close to the cluster center is designated as cluster head (CH). CH manages the locations of nodes belonging to that cluster. While moving within a cluster, a node needs to propagate its location only to the nodes that reside in that cluster. A node's cluster position is updated when it leaves its own cluster and enters another cluster. Routing is performed using two steps. The first step is global geographic routing in which packets are routed from one cluster to another cluster that is closer to the destination depending on cluster positions. Messages are forwarded based on right-hand rule in case of dead end cluster (i.e., there are no clusters closer to the destination cluster than the current one). The second step is called local gradient routing in which packets are routed between clusters. A CH broadcasts a tree-building message to its cluster nodes and to all the nodes covered by its neighboring clusters. After the message reaches all neighbor CHs, nodes would have set up a route to its cluster head and the cluster heads of its neighboring clusters.

When a source node sends a packet to a destination node, it sets the destination's cluster position (DCP) field of the packet. Any node U receiving the packet checks the DCP field to see whether its cluster is the intended one or not. If its cluster is the intended one, U sends the packet to its cluster head based on local gradient routing. Then the packet is routed to the destination node by the cluster head. Otherwise, U determines the next cluster position (NCP). A next cluster is selected from the neighboring clusters through global geographic routing scheme. This process continues until the destination cluster is reached if there is a path, or the packet is dropped.

2.1.4 Localized Load-Aware Geographic Routing in Wireless Ad Hoc Networks

Li et al. [75] proposed localized load-aware geographic routing based on the concept of cost-to-progress ratio in greedy routing (CPR-Routing). Load could be defined according to the resource availability. It may be storage capacity, processor usage, traffic amount, link quality, power consumption, and/or activity. In this paper, the authors use communication activities for defining load. A load limit L is dedicated for each node. Each node's load L is initially zero and then incremented by specific units for the node and by other units for its neighbors. The main idea behind this protocol is to combine the greedy forwarding technique and localized cost-to-progress ratio (CPR) [52] framework proposed by Stojmenovic. In greedy forwarding, the node that is closer to the destination than the relaying node is selected as the next hop. CPR-routing uses an objective function that involves both distance and appropriate load metric as a criteria to make next hop decisions. This technique tries to avoid two main problems, namely, the dead end problem of greedy forwarding and the probability of overloading nodes (nodes reaching their load limits). In addition to location information, to route around areas containing nodes with high load, nodes need to know the most recent load information of themselves and their neighbors. For that purpose, *HELLO* messages are used to obtain neighbors' load information.

CPR-Routing assumes that each edge in the network has an associated cost. The node N holding the packet uses the objective function F_{crp} which is cost of ratio progress defined as follows:

$$F_{crp}(A) = \frac{Cost(NA)}{|ND| - |AD|}$$

Here, node A is one of N's neighbors. $Cost(NA)$ is $\mathit{Load}(A)$, the load of node A. $|ND| - |AD|$ is the progress which measures the oncoming of A towards D. This function is applied on every neighbor of node N. The neighbor A with the minimum value of F_{crp} is chosen as the next hop. This process is repeated at every node until the packet reaches final destination.

2.1.5 Local Area Network Dynamic Routing Protocol: A Position Based Routing Protocol for MANETs

The local area network dynamic routing protocol (LANDY) [76], a position based routing protocol for MANETS by Macintosh et al., claimed to be a low overhead, light-weight, routing protocol. LANDY works as follows:

- *Locomotion Prediction of Mobile Nodes* LANDY uses locomotion (movement) and velocity of each node to predict the future location of each of these nodes so that data packets can be forwarded efficiently towards the destination nodes. If the forwarding process fails due to obstacles or dead ends, a perimeter routing recovery procedure similar to that used in GPSR [8] is used to heal the problem. A HELLO message contains the mobile node's locomotion component (LC). The LC contains the mobile node's unique code identifier (MCID), its cell unique code identifier (CCID), its three sample positions (p1,p2,p3), velocity, and three different time stamps of the three sample positions.

- *Forwarding Protocol* Each node maintains a locomotion table (LT) which is updated on receiving HELLO from its one-hop neighbors. Depending on the LT, the source node forwards the data packet to the next hop N which will be closer to the destination in future.

- *Detecting Failure and Recovery Process* A data packet can be in two modes, forwarding mode or recovery mode. The format of the data packet contains the mode (forwarding or recovery), hop count, destination LC, LC of the node where the packet entered recovery mode, LC of previous node, and life time. When a node encounters local maxima, the packet enters a recovery mode. LANDY stores that position where the packet has entered the recovery mode. Then the packet is forwarded based on right hand rule along the first adjacent edge in the planar graph of the network constructed. Upon receiving a packet that is in recovery mode, each node probes to see if it is the intended destination. If not, it tries to figure out whether it can recover from recovery mode. If yes, the packet returns to the greedy forwarding mode. Otherwise, the packet traverses along the edges of the planar graph in the recovery mode.

Since this protocol uses locomotion feature to predict the future position of the mobile nodes, it can help in predicting location of the destination more accurately and helps in more efficient packet delivery.

2.2 Hybrid Routing Protocols

In this section, we present a summary of the basic idea behind several of the recently proposed hybrid routing protocols that depend either directly or indirectly on nodes' position information for routing.

2.2.1 An Anchor-Based Routing Protocol with Cell ID Management System for Ad Hoc Networks

Anchor-based Routing Protocol with Cell ID management system for MANETs (ARPC) [16, 77], proposed by Li and Singhal provides routing information for nodes that desire to communicate with other nodes. The routing process is divided into the following five main parts:

- *Location-based Clustering Protocol* Several physical locations (e.g., some known buildings in a campus region) in the network area are assumed to be known. These known locations are called anchors which have assigned coordinates. The network area is partitioned into cells. Each cell will have an anchor that is set as the cell's center. It is assumed that there is always a node close to each anchor, this node is called the agent of that anchor. The authors assume that each node knows the coordinates and IDs of all anchors. Each anchor's agent periodically broadcasts a message announcing its anchor ID to the nodes of the cells with one or two hop counts. Upon receiving the announcement, a node joins the cell from which the announcement came and takes the anchor ID as its cell ID. Since each node knows the IDs of all the anchors, each node can recognize the neighboring cells. When an agent of an anchor moves away, a node close to the anchor of the cell is selected as the new agent of the node.

- *Inter-cell Routing Protocol* This part lets every node maintain a dynamic routing table that contains routes to its neighboring cells. A bridge node is the first reachable node in a cell from a neighboring cell's node. Each entry in the routing table represents an available path to one of the neighboring cell's bridge node. A source node generates RREQ messages only when no routes to the neighboring cells are found. To find routes to the neighboring cells, a source node broadcasts a route request message (RREQ) to its neighboring cells. Since the node knows the cells' IDs, it includes the IDs of the intended neighboring cells in the RREQ message. When an intermediate node, a node within the same cell, receives the RREQ packet, it searches its routing table for paths to the intended neighboring cells. If it has paths, it creates a route reply message (RREP) containing all the known bridge nodes in the neighboring cells and unicasts the message back to the source node. If the intermediate node does not have paths to the neighboring cells, it rebroadcasts the RREQ message. When a node in an intended neighboring cell receives the RREQ message, it becomes a bridge node, creates an RREP message, and unicasts it back to the source node.

- *Intra-cell Routing Protocol* This protocol uses on-demand routing approach for routing packets inside the same cell, similar to AODV [15].

- *Data Packet Routing* This stage explains how the data packet is transmitted from a source to a destination node. The source node first looks up its inter-cell routing table for the destination's cell ID. If it does not have the destination's cell ID, then it checks its intra-cell routing table for a route to the destination. If there is a route to the destination in the intra-cell routing table, it uses intra-cell routing mechanism to deliver the data packet. Otherwise, the destination is in another cell. In this case, the node consults the Cell ID Management System to get the destination's cell ID. If the destination is in different cell, the neighboring anchor that is closest to the destination anchor is chosen as the next hop. Note that each node has built up its inter-cell routing table using inter-cell routing protocol. Therefore, the source node forwards the packet to the bridge node of the next anchor (next hop). Upon receiving the data packet, the bridge node applies the same procedure until the destination cell is reached. When the destination cell is reached, the bridge node delivers the packet to the destination node using the intra-cell routing technique mentioned above.

- *Cell ID Management System* Nodes' cell IDs may change due to node movement. Therefore, the system needs a cell ID management system to handle this. Each anchor's agent manages the cell IDs of all the other nodes belonging to the same cell in which the agent resides. This makes the agent store and maintain addresses and cell IDs of the other nodes in a table called CELL-ID-TABLE. All the agents together manage the Cell IDs; it is a distributed database system that has the agents as its servers. When a node N needs to find its agent, it computes the mod of the node ID of N with respect to the total number of cells in the network. The residue of this mod represents the cell ID whose agent is responsible for storing and maintaining the cell ID of node N. After obtaining the cell ID through Location-Based Clustering phase and during the network setup process, a node generates a cell ID registration message CELL-ID-REGISTRATION. This message contains the cell ID of the agent managing the cell ID of that node, the address of that node, and the cell ID of that node. The node sends the CELLID- REGISTRATION to its agent using the data packet routing protocol mentioned previously. Upon receiving the message, the agent inserts the node's address and cell ID in its Cell-ID-Table. When a node leaves its current cell and enters another cell, it needs to update its status in its server (agent). The agent then updates cell ID of that node. It is possible that the agent moves to another cell. In that case, the agent should send its Cell-ID-Table to a newly designated agent which takes over the agent's role.

2.2.2 Direction Assisted Geographic Routing for MANET

Zhou et al. [78] presented the Direction Assisted Geographic Routing (Geo-DFR) for MANETs. Geo-DFR incorporates directional forwarding in routing (DFR) [79]. Routing is done mainly using greedy forwarding. However, in case of dead ends DFR is used. DFR [79] was mainly designed to solve the corrupted next hop problem resulting from outdated routes in routing tables. Geo-DFR improves the original DFR to solve the dead end problem so that perimeter face routing is avoided. The authors use Fisheye State Routing protocol (FSR) [80] as the hosting protocol for Geo-DFR. FSR is a proactive routing protocol based on Distance Vector (DV) routing. Each remote destination node broadcasts routing updates periodically using different frequencies when it receives a route request from a source node. The destination node's location information as well as other necessary information are embedded in the update packet. In addition to that, every node locally floods its topology information for a maximum of k hops by using proactive Link State (LS) routing protocol. The routing updates include the piggybacked neighbor coordinates. This provides each node with accurate routing information as well as coordinates of all local nodes up to k hops. To reduce the local update flow, k is set to 2 in Geo-DFR.

Geo-DFR maintains the following three tables at each node.

1. *Neighbor Coordinate Cache Table,* which maintains coordinates of all the neighbors within its local scope.

2. *Direction Cache Table,* which maintains the directions and positions of remote destinations through propagating Distance Vector routing updates. This table is refreshed periodically and its entries expire after a predefined amount of time. The refresh time normally depends on nodes' mobility rate.

3. *Local Routing Table,* which is created and maintained by the proactive local scoped routing protocol, offering accurate local routing information to nodes inside the local scope.

The direction of the predecessor node is calculated based on the coordinates of both the current node and the predecessor. In addition to keeping the predecessor, Geo-DFR traces the direction of the routing update that has arrived from the destination node. In other words, Geo-DFR gets the direction to which data packets are forwarded from routing updates, and not from destination's coordinates. This direction assists in deciding the next hop to forward packets. The greedy mode of the protocol is utilized if the calculated direction (next candidate's direction) and the greedy direction are consistent or they differ which helps in early detection of dead ends. Otherwise, packet forwarding is done using DFR. This gives the protocol both the geographic routing feature and the topology-based routing feature.

For forwarding a packet, the source node consults its local routing table for a route to the destination. If there is an existing route for that destination in the table, the packet is routed accordingly. Otherwise, the destination is outside the local scope and is considered as a remote destination. This makes the source node invoke (broadcast) an on-demand route discovery operation. When the destination receives the route request, it starts to periodically broadcast beacon messages to its direct neighbors, announcing its existence. Upon receiving a beacon message, neighbor nodes update their routing information (i.e., the next hop ID and the direction to the destination) and in turn, inform their neighbors of the update they received. The proactive beaconing started by the destination ends after a predefined time within which no data packets have been received by the destination. This may be because the source node has already finished transferring its data. In case of dead-end or no next-hop for the destination, the direction cache table can be consulted to forward the data packets. When they get close to the destination node, the data packets are forwarded using the local routing table. According to the above discovery procedure, the source node and/or any intermediate node can consult its cache table or its local routing table to get the destination's coordinates and directions. Therefore, based on this information, packets are forwarded either greedily or using the DFR mode of routing.

2.2.3 A New Hybrid Location-Based Ad Hoc Routing Protocol

Al-Rabayah et al. [81] present a New Hybrid Location-based Ad Hoc Routing Protocol (HLAR) that mainly addresses scalability. HLAR is a combination of both AODV [15] and an expected transmission count (ETX) metric based protocol [82] which reduces the total number of packet transmissions and retransmissions. The main purpose of this combination is to discover an optimum quality route metric called AODVETX. This metric efficiently utilizes all possible available location information, thereby reducing the control overhead used to establish and maintain routes. This feature allows intermediate nodes to fix link breaks instead of merely reporting the problem back to the sender. This protocol works as follows.

- *Protocol Work-flow* Initially, all the nodes exchange their location information and IDs with their neighbors through beacon messages to build their neighbor tables. When a source node needs to send data packets to the destination but it does not have a route to the desired destination, it tries to find a route to the destination in an on demand fashion. As mentioned above, this process is done geographically by using the greedy forwarding technique. The source puts the destination's location information in addition to its own location information in a route request packet RREQ. Then it tries to greedily find a neighbor from its neighbor table that is closer to the destination node. If a closer node is found, the source forwards the RREQ packet to that node; otherwise, there may be a dead end (void) or neighbor nodes have no location information from which distances to the destination can be found. In this case, the relaying node floods a RREQ packet to all its neighbors. Rebroadcasting packets is set to a maximum of TTL times. The time-to-live (TTL) field is set by the source of the request based on the estimated number of hops between the source and the destination. Each time a node faces a dead end or a void, it decrements the TTL value by one; when TTL reaches zero, the rebroadcasting process stops. Upon receiving the RREQ packet, the destination node responds with a route reply message RREP if one of the following three conditions is true.

1. It is the first time that this RREQ has been received.
2. The currently received RREQ has greater source sequence number than the previous received RREQ.
3. The RREQ has the same source sequence number as the previously received RREQ's sequence number, but it shows that a better quality route exists.

- *Repairing Functionality* The repairing functionality of the protocol works as follows: when an intermediate node determines a broken link towards the destination, it stores the received data packet locally in its buffer. Then it consults its neighbor table to see if it has a neighbor with shorter distance to the destination node; if yes, it updates its routing table and forwards the data packets to that neighbor node. If there is no neighbor node closer to the destination than itself, it broadcasts a route repair packet RRP to its neighbor nodes and sets the TTL field of the RRP packet to the remaining number of hops to the destination from its location. Similar to the route setup process mentioned above, a neighbor node checks its routing table to see whether it has an up-to-date route to the intended destination; if so it replies with a route repair reply packet RRRP to the intermediate node. If a neighbor does not have an entry to that destination in its routing table, it tries to find one of its neighbor nodes that is closer to the destination than itself. If it has one, it replies with a RRRP packet to the intermediate node. Otherwise, it floods its neighbors with a RRP packet and decrements the TTL by one. This process continues until the destination is reached or the TTL becomes zero, in which case, the packet is dropped. When the destination is reached, similar to the main route set up process, the destination sends a route reply to the intermediate node so that that intermediate node can continue forwarding the data packets without going back to the main source node about the route break.

2.3 Discussion and Analysis of Reviewed Protocols

In this section, we discuss the merits and demerits of the geographic and hybrid routing protocols we discussed in Sections 2.1 and 2.2, respectively. The discussed protocols mainly differ in the way in which they handle the dead end problem. Other issues taken into consideration in designing these protocols are: scalability, reducing control overhead, load balancing, fault-tolerance, robustness, and reducing complexity. Depending on how well the discussed protocols address these issues, we rank the protocols low, Medium, and High with respect to these parameters as shown in Table 2.1. Next, we present a detailed performance comparison of these protocols with respect to these parameters.

Table 2.1: Comparison of the Surveyed Protocols with respect to Different Features

Protocol	**Scalability**	**Overhead**	**Fault-Tolerance**	**Robustness**	**Complexity**
HIR	Low	Medium	-	High	High
LGR	Medium	Low	High	Medium	High
ARPC	Low	Medium	-	Low	Low
Geo-DFR	Low	Medium	-	Low	High
CPR-routing	High	Low	-	High	Low
HLAR	Medium	Low	-	Medium	Low
LANDY	Medium	High	Medium	High	High
GPSR	High	High	-	Medium	High

2.3.1 Scalability

Designing scalable routing protocols for MANETS is a challenge. A protocol is scalable if it continues to perform well as the number of nodes in the network increases. HIR [2], does not scale well due to the central packet flooding technique used by a designated node to select LANDMARKs. The packets will be received by every node in the network and each node participates in making the LANDMARK election decision. This process limits scalability. However, after selecting the LANDMARKs, data packet forwarding will be faster since sender nodes use the LANDMARKs that have been established previously. So, HIR is moderately scalable. In LGR [6], there is one flooding technique. A cluster head (CH) broadcasts messages to all the nodes in its own cluster as well as all nodes in its neighboring clusters so that each of these nodes can establish a route to the CH. This approach does not scale well as the number of nodes in the network increases. This is because if nodes are highly mobile, frequent CH election occurs which results in high message overhead. LGR is more scalable than HIR since LGR does not broadcast control packets or data packets to the entire network.

ARPC[16] is less scalable since nodes use routing tables which can become large as the network size increases. Moreover, the announcement packets sent by agent nodes to announce their existence to the nodes in the cell they reside in can cause large overhead as the network grows. When no route is found in the nodes' routing tables, there is a possibility of two level broadcasting of messages, namely, Inter-cell broadcasting, which occurs across all cells in the network and, Intra-cell broadcasting which happens inside a cell. The second one has lower effect on scalability than the first one. In both Inter-cell broadcasting and Intra-cell broadcasting, rebroadcasting may occur until the intended node is reached. So, ARPC is less scalable than LGR. Scalability in Geo-DFR [78] is affected by different factors. First, maintaining three tables in each node increases overhead especially when a node has many neighbors. But this limitation is local, since the number of the records in two of these tables depends on the density of the neighboring nodes. The other table (Direction Cache Table) holds destinations' directions and coordinates. However, records expire after a predefined time which helps in reducing the table size. When information found in the tables are out of date, the protocol depends on two flooding operations which affects the scalability of the algorithm. These factors make this protocol to be worse than the protocols discussed so far.

Since CPR-Routing [75] protocol uses only local broadcasting, its scalability is limited only to a local scope which means that unless there are a large number of local (neighboring) nodes, the protocol scales well. We can note that CPR-Routing is the most scalable protocol among all the above mentioned protocols since the information exchanged between nodes to route data packets is much less than the other protocols. HLAR [81] is another protocol that is scalable which using periodic broadcast messages, keeps routing information in tables so that less routing requests are broadcast when source nodes need to send packets. In spite of that, source nodes may need to broadcast route request packets when there are no routes to the destination in their Neighbor Tables. In that case, the broadcasting process is limited to a pre-determined number of hops (TTL) set by the source node. Even though this protocol is proposed to address scalability issue, it still utilizes large number of message exchanges to set up routes. However, CPR-Routing exchanges less messages and it provides higher scalability than this protocol. LANDY [76] is a scalable protocol since it has only local broadcasting to build a Locomotion Table (LT). Sending data packets only depends on LT and there is no global broadcasting in case of failure or dead ends. Instead of broadcasting, when forwarding fails, a recovery mode is invoked from the point of failure; hence allowing the protocol scale better.

The above scalability analysis of the discussed protocols gets us to conclude the following. Broadcasting for route requests and route maintenance is one of the main operations that can determine scalability. We notice that each one of the discussed protocols uses broadcasting technique, but in different ways. Each one of them uses local broadcasting to connect the nodes that are within the transmission range of each other. This broadcasting can limit scalability but only in local scope. Since all the protocols use this technique, they will share the same characteristics and will have the same limitation with respect to scalability. Because of that, we try to consider global broadcasting as well as other factors that can affect scalability. These factors may be the amount of messages exchanged to setup routes, size of packets, and/or routing tables. Most of the above mentioned protocols utilize global broadcasting, each in different way. Considering all of these factors we can sort the protocols with respect to scalability, from the most scalable to least scalable as follows: CPR-ROUTING, GPSR, LANDY, LGR, HLAR, HIR, ARPC, and GEO-DFR.

2.3.2 Control Overhead, Complexity, Latency, and Robustness

Other factors that need to be taken into consideration while designing routing protocols for MANETs are Control Overhead, Complexity, Latency, and Robustness. Control overhead measures the amount of control information that are exchanged between nodes before starting and/or during the data packet forwarding process. Overhead on nodes also includes the amount (size) of information that nodes can use to build routing tables. More control overhead generally results in more successful packet delivery, but may not be scalable. Robustness, on the other hand, measures the degree of successful data packet delivery. Highly robust protocols guarantee that packets are delivered to their destinations if there is a route from source node to destination. Complexity shows the amount of calculations required by the protocol to build routing tables, make routing decisions, and/or forward data packets. Latency measures the speed at which packets are delivered. Faster the packets arrive at their destinations, lower the latency is.

In HIR [2], nodes proportionally exchange high amount of control information to select LANDMARKs. However, since this happens only to select LANDMARKs, normal data packet forwarding process does not incur additional overhead. On the other hand, the level of stability of LANDMARK nodes plays an important role in the overhead in constructing LANDMARKs. More stable LANDMARKs results in less control overhead. So, overhead in this protocol mainly depends on the rate at which nodes move and the network size. On the other hand, up-to-date maintenance of LANDMARK nodes helps nodes forward packets correctly. In addition to that, using Expanding Ring Flooding when both, the Greedy mode and the Detour mode fail makes it more robust. Beside that, the switching technique from Greedy to Detour mode and from Detour to Greedy mode makes the protocol loop free which gives better robustness.

LGR [6] deals with dead end problems using cluster-based techniques. Electing Cluster Heads (CH) overwhelms nodes with control packets exchanged between nodes. Choosing new CHs when a CH leaves its cluster means that nodes must compare their distances again to elect one of them as a CH, for their cluster. The new CH then announces its existence to the other related nodes within its cluster and its neighboring clusters. This results in longer end-to-end delay and increased overhead. These calculations incur reasonable complexity which is lower than the complexity produced by HIR. However, LGR can make use of these information and produce better robustness. In addition to that, using Right Hand Rule to overcome the dead end problem faced by the Greedy mode can give more robust data packet forwarding. We notice that the overhead in this protocol is similar to or less than that of HIR, but HIR is more robust than LGR since HIR has more concrete alternative solutions to voids at the cost of more complexity than the complexity of LGR.

In ARPC[16], overhead is high on nodes since each node maintains two routing tables, namely inter-cell routing table and intra-cell routing table. This is obviously useful for better data packet forwarding since routing information are mostly available in the tables giving better performance, lower complexity, and lower end-to-end delay. On the other hand, considering the available information and anchors, the protocol simply uses Greedy forwarding that may fail in case of dead ends or voids. However, the protocol does not use any recovery mechanism to cope with this problem; hence ARPC is not robust.

Maintaining three tables for each node in Geo-DFR [78] results in high overhead. Finding and updating the records of the Direction Cache Table as well as the other two local tables results in more complexity. The protocol ensures better performance and lower end-to-end delay since packets are forwarded mainly in Greedy mode. CPR-Routing [75] uses reasonable information to make routing decisions which reduces overhead in routing using an objective function for selecting forwarding node. This means that data packets are forwarded directly without building routes or maintaining routing tables in nodes. A drawback of this approach is the complexity involved in the calculation for selecting appropriate neighbor to forward a packet which could result in higher end-to-end delay. The end-to-end-delay could also increase when the protocol selects route that has lower cost. Using this technique, the protocol has potential to avoid the dead end problem since packets are routed choosing the best route determined based on the objective function rather than purely Greedy forwarding, producing robust routing. HLAR [81] uses only neighbor tables in nodes which depends on the network size. Since intermediate nodes participate in repairing broken routes, latency will be lower, and has lower complexity. Robustness can be affected due to intermediate nodes buffering data packets until a broken route is repaired because packets could be lost due to buffer overflow.

Overhead involved in LANDY [76] is higher than that of the protocols discussed so far since it uses more control information to build tables at each node. That includes different samples of each node's location information, exchanged periodically between nodes, building planar graphs to be used as alternatives to the normal forwarding mode. On the other hand, the protocol can greatly make use of this information to successfully deliver data packets to destinations which makes the protocol more robust with lower end-to-end delay. That is because routing decision of packets are taken locally when dead ends are faced which takes less time. This makes the protocol more robust than the previously discussed protocols but at the cost of more overhead and complexity than the other protocols.

2.3.3 Load Balancing and Fault-Tolerance

Next, we compare the Load Balancing and the Fault-Tolerance aspects of the surveyed protocols. Load Balancing refers to spreading the data packets that need to be sent over multiple routes without overwhelming a node or a group of nodes and leaving the others unused. This feature makes packets travel faster throughout the network. Fault-Tolerance relates to how efficiently a protocol deals with incorrect location information of the destination nodes for successful delivery of the packets. The discussed protocols do not provide specific strategies to deal with the load balancing. However, some of them (e.g., CPR-Routing [75]) implicitly provide a moderate level of load balancing. CPR-Routing tries to balance load by using nodes' load and choosing the next hop that satisfies a load condition. This means that each time a packet is forwarded, different nodes are selected as next hop based on a predefined metric which gives the chance of spreading out the packets over more nodes producing medium level of load balancing.

Fault-Tolerance is achieved by the surveyed protocols, namely LGR [6] and LANDY [76]. The other protocols assume that location information are correctly provided. However, LGR simply routes data packets using fixed cluster positions instead of accurate nodes' positions, so inaccurate location information does not affect the process. LANDY uses three position samples of each node. This helps having better estimate of nodes' locations especially when the velocity of the sampled nodes are known. However, we can see that LGR tolerates faults better when compared to LANDY. This is because LGR does not use nodes' locations to route packets while LANDY does that, but reduces the chance of having inaccurate location information.

2.4 On-demand Routing Protocols

In this section, we review related on-demand routing protocols bringing out their merits and demerits.

2.4.1 Ad Hoc On-demand Distance Vector Routing (AODV)

Under AODV [15], when a node needs to establish a route to a destination, it broadcasts a route request to all its neighbor nodes. A node receiving the route request replies to the source node, if it has a route to the destination; otherwise it rebroadcasts the route request to all its neighbors. This process continues until the a route to the destination is found. This protocol is robust because broadcasting a route request guarantees finding a route to the destination if there is one; however, as number of nodes increases, the number of redundant rebroadcasting of route requests increases. This means this protocol is not scalable.

2.4.2 Dynamic Source Routing (DSR)

DSR [33] is an on-demand routing protocol in which a source S initiates a route discovery to find a route to the target T. In the route discovery phase, the source broadcasts a route request to all its neighbors. A route request contains a source identifier, a target identifier, a request identifier, and a path (i.e., a route). On receiving the route request for the first time, each intermediate node N appends its identifier to the node list in the request (i.e., to the path) and rebroadcasts the request. When the route request arrives at the target T, this target creates a route reply which contains the path received in the request (i.e., the accumulated list of nodes). The target then forwards the reply through the accumulated list of nodes in the reverse order. When the source S receives the reply, it creates a new record in its Route Cache that contains the route returned in the reply.

Whenever a node transmits a data packet, a route reply packet , or a ROUTE ERROR packet, it must verify that the next hop correctly receives that packet. Because DSR is a source routing protocol, when an originator sends a data packet, it inserts the complete route into the header of that data packet. Each intermediate node N along with the route forwards the data packet to a next hop H indicated in the header of the data packet. N then verifies that H has received the packet.

However, if the intermediate node N cannot make the confirmation within a limited number of local retransmissions of the data packet, N returns a ROUTE ERROR packet to the source of that data packet. This ROUTE ERROR packet indicates that the link from N to the next hop H is broken. The source then deletes the broken link from its Route Cache and tries another route if one cached or restarts the route discovery. The redundancy of the route requests and the accumulated nodes in these requests make this protocol non-scalable and result in high overhead.

We propose an on-demand protocol, that lets the route requests propagate using a greedy approach until the destination is reached. This helps in reducing the overhead and route requests are not broadcast under our protocol. In addition to that, the route requests traverse the shortest path possible because they are propagated using a greedy approach.

2.5 Secure Routing Protocols

In this section, we review some of the source routing protocols, presented in the literature which were claimed to be secure.

2.5.1 Secure Routing for MANETs (SRP)

SRP [46] is an on-demand source routing protocol that has the basic characteristics of reactive routing protocols. Route requests in this protocol are generated by a source S and protected by Message Authentication Code (MAC) [83]. The MAC is computed using the key that S shares with the target T. S broadcasts the route request to all its neighbors. When each neighbor of S (or an intermediate node) receives the route request, it appends its id to the request and rebroadcasts the updated request if it has not seen that request ever. Otherwise, it discards the request. Intermediate nodes do not check the validity of the MAC in the request because no node possesses the key used to compute it except S and T.

When the target T receives the request, it verifies the MAC in the request. If the MAC is valid, then the target assumes that all the adjacent pairs of nodes accumulated in the route request are neighbors. This route is considered a valid or a plausible route. The target then computes a MAC that authenticates the route using the key it shares with the source. This is then sent back to the source S through the reverse route traversed by the request. For example, a route request message received by an intermediate node X_j has the following form

$$m_{rreq} = (rreq, S, T, id, sn, (X_1, ..., X_j), mac_S),$$

where id is a route id that is randomly generated, sn is a sequence number, and mac_S is the MAC computed on $rreq,\ S,\ T,\ id$, and sn by S using the key it shares with T. Now, if $S,\ X_1,\ ...,\ X_p,\ T$ is the discovered route, then all intermediate nodes X_j, $1 \leq j \leq p$ will receive m_{rrep} as a route reply, where

$$m_{rrep} = (rrep, S, T, id, sn, (X_1, ..., X_p), mac_T),$$

where mac_T is the MAC computed on the message fields preceding it by the target T with the key it shares with S. Intermediate nodes must check the route reply header to verify that both id and sn fields are correct. In addition to that, intermediate nodes need to check that they are neighbors with both their upstream and downstream nodes before sending the route reply upstream.

An intrinsic point to observe in this protocol is that the upstream route from T

to S is authenticated by the target, but the downstream route from S to T is not authenticated by S. This means that there could be malicious pair of nodes that are not neighbors but claim to be neighbors in the route request that reaches the target. These malicious nodes might divert traffic through other routes as observed in [46]. It is also possible for a malicious node to pad route requests with identifiers that are not its neighbors. The malicious node can impersonate these nodes in the reply phase in such a way that the reply propagates to the source node. Hence, the route received by the source may be invalid because some of the nodes specified in the route as neighbors may not be neighbors.

2.5.2 The Basic Ariadne Protocol

Hu et al. [21] presented ***Ariadne***, a secure source routing protocol, based on Dynamic Source Routing protocol (DSR) [33]. When a source S performs a route discovery for a target T, it is assumed that both S and T share the secret key K_{SD} and K_{DS} for authenticating messages. Ariadne imposes two requirements in the route discovery phase:

- First, that a source node can authenticate each node in the route (i.e., the node list in the route reply).

- Second, that the target can authenticate each node in the route traversed by the route request so that it returns a route reply along a path that has legitimate nodes.

To achieve node list authentication, the authors use three different techniques: the TESLA protocol, the standard MACs, and the digital signatures. TESLA [84] is a broadcast authentication scheme that requires time synchronization. Unlike other asymmetric protocols such as RSA [85], TESLA achieves the asymmetry from both clock synchronization and delayed key disclosure. However, TESLA depends on the ability of a receiver to correctly ***determine which keys a sender might have published (i.e., the TESLA security condition)***.

In the following subsections, we discuss the three techniques used in Ariadne mentioned above and present the attack found on each one of them.

2.5.3 Basic Idea behind Ariadne with Signature

Ariadne with signatures proposed by Hu et al. [21] differs from SRP in two main aspects as noted in [4]. First, in addition to source and target nodes, intermediate nodes include their own digital signatures in route requests. Second, per-hop hashing is used to prevent removal of legitimate nodes from node lists in route requests. A source node broadcasts a route request message to its neighbors. The route request contains the source id and the target id, a random request id, and a MAC computed over these elements using a key that the source shares with the target. Together with its own id, each intermediate node hashes the MAC using a one-way hash function. These hash values computed by intermediate nodes are called per-hop hash values. The intermediate node then appends its id to the node list accumulated in the request and generates a digital signature over this updated request. This signature is then appended to the signature list in the request and the request is re-broadcast. For example, as noted in [4], a route request message forwarded by an intermediate node X_j has the following form

$$m_{rreq} = (rreq, S, T, id, h_{X_j}, (X_1, ..., X_j), (sig_{X_1}, ..., sig_{X_j})),$$

where S and T are the source and target identifiers respectively, id is the random request identifier, h_{X_j} is the per-hop hash value calculated by X_j, $(X_1, ..., X_j)$ is the node list, and $(sig_{X_1}, ..., sig_{X_j})$ is the signature list. When the route request arrives at the target T, T verifies the MAC of the source, the per-hop hash of each intermediate node, and individual signature of each intermediate node in the signature list. If these verifications are successful, the target generates a route reply and sends it back through the list of nodes in the request in the reverse order. Each intermediate node forwards the reply to the next hop without any modification of the reply. The reply contains the source id and the target id, the accumulated route and the digital signatures of the intermediate nodes obtained from the route request, and a digital signature that the target computed over these elements.

When the source S receives the reply, it verifies the digital signature of the target and the individual signature of each intermediate node. S accepts the route returned in the route reply if all these verifications are successful.

2.5.4 Basic Idea Behind Ariadne with MAC

Ariadne with MAC presented by Hu et al. [21] is similar to the one presented in Section 2.5.3 with the exception that it does not use digital signatures. In the route discovery phase, a source S broadcasts a route request message intended to find a route to a target T. The request message contains the ids of both the source and the target, a random request id, and the MAC that is computed over these elements. The MAC is computed by S using a key (K_{ST}) it shares with T. When an intermediate node X receives this MAC, X hashes the MAC with its own id using a one-way hash function. These hash values are called per-hop hash values which prevent the removal of node ids from the node list in the route request.

When an intermediate node receives the request for the first time, it computes the per-hop hash value, appends its id to the accumulated node list contained in the request, and computes its own MAC on the updated request using the key it shares with T. The intermediate node then appends its MAC to the MAC list in the route request and re-broadcasts the updated request. For example, as observed in [3], a route request message m_{rreq} received by an intermediate node X_j has the following form

$$m_{rreq} = (rreq, S, T, id, h_{X_j}, (X_1, ..., X_j), (mac_{X_1}, ..., mac_{X_j})),$$

where S and T are the source and target identifiers respectively, id is the random request id, h_{X_j} is the per-hop hash value (i.e., hash chain) calculated by X_j, $(X_1, ..., X_j)$ is the node list, and $(mac_{X_1}, ..., mac_{X_j})$ is the MAC list, where

$$h_{X_j} = H(X_j, H(X_{j-1}, H(..., H(X_1, h_S)))),$$

h_S is the hash value computed by S over the main fields of the route request. i.e.,

$$h_S = MAC_{K_{ST}}(S, T, id).$$

When the target receives the route request, it verifies the hash value and MAC attached by the source and verifies the hash value and MAC attached by each intermediate node. If these verifications are successful, the target generates a route reply and unicasts it back via the reverse route obtained from the route request. The route reply contains the id of the source and target, the node list obtained from the request, and a MAC computed by the target over these elements. The MAC attached by the target is computed using the key shared by the target and the source. Intermediate nodes do not modify the reply. Route reply m_{rrep} created by the target T is of the following form

$$m_{rrep} = (rrep, S, T, (X_1, ..., X_j), mac_T).$$

When the source S receives the route reply m_{rrep}, it only verifies the target's MAC mac_T. S accepts the route returned in the reply if this verification is successful. Otherwise it discards the reply.

2.5.5 Basic Idea Behind the Optimized Version of Ariadne with Iterated MAC

Ariadne has another version that uses iterated MAC computations [86] instead of independent MACs that are computed separately,as in Section 2.5.4. As noted in [3], compared to the other versions of Ariadne, this iterated MAC version has superior security characteristics and is more secure. Like the other versions, a source node broadcasts a route request to all its neighbors. Each intermediate node updates the request that is received for the first time and re-broadcasts the updated request. A route request that reaches an intermediate node X_j, $1 \leq j \leq p$, on a route $S = X_0$,X_1,...,X_p,$X_{p+1} = T$ is of the following form

$$m_j = (rreq, S, T, id, (X_1, ..., X_j), mac_{SX_1 \cdots X_j}).$$

Where id is a randomly generated request id, $(X_1, ..., X_j)$ is the accumulated route (i.e., the node list), $mac_{SX_1 \cdots X_j}$ is the MAC computed by X_j with the key it shares with the target T over the route request m_{j-1} received from X_{j-1}, where

$$m_{j-1} = (rreq, S, T, id, (X_1, ..., X_{j-1}), mac_{SX_1 \cdots X_{j-1}}).$$

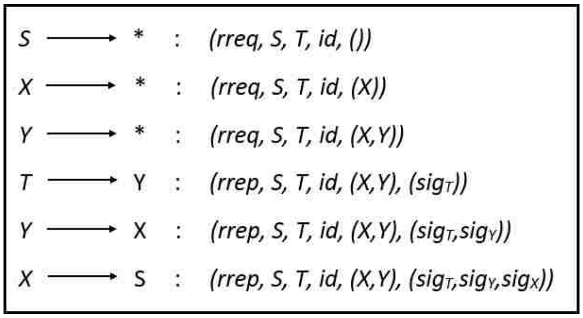
When the target T receives the route request from the last intermediate node X_p, it recomputes all intermediate MAC values. T shares a key with each intermediate node, therefore, it iteratively reconstructs the MAC sequence and compares each value with the corresponding value it computed up to the last value that should match the MAC received from X_p. If the verification is successful, it means that each intermediate node in the node is genuine. The target T then generates a route reply of the form

$$m_{rrep} = (rrep, S, T, id, (X_1, ..., X_p), mac_T),$$

where mac_T is the MAC computed by T with the key it shares with S on the message fields that precede it (i.e., on $(rrep, S, T, id, X_1, ..., X_p)$). The reply m_{rrep} then is unicast via the nodes in the request in the reverse order (i.e., via the nodes $X_p, X_{p-1}, ..., X_1$) to the source S. When and intermediate node receives the reply, it verifies that its id is in the node list. It also verifies that the id preceding it and the id next to it in the node list are its neighbors. However, intermediate nodes do not modify the reply. When the source receives the route reply m_{rrep}, it accepts the route returned in m_{rrep} if it can successfully verify the MAC computed by the target, i.e., mac_T.

2.5.6 Basic Idea Behind endairA Protocol

$\grave{A}cs$ et al. presented endairA [3], a secure source routing protocol, in which the route reply is authenticated; hence intermediate nodes sign the route reply instead of the route request. A source S broadcasts a route request that contains the identifiers of



S → * : (rreq, S, T, id, ())
X → * : (rreq, S, T, id, (X))
Y → * : (rreq, S, T, id, (X,Y))
T → Y : (rrep, S, T, id, (X,Y), (sig_T))
Y → X : (rrep, S, T, id, (X,Y), (sig_T,sig_Y))
X → S : (rrep, S, T, id, (X,Y), (sig_T,sig_Y,sig_X))

Figure 2.2: An Illustration of the Operation of endairA. S is the source, T is the target, and X and Y are the intermediate nodes. id is a request id that is randomly generated. sig_X, sig_Y, and sig_T are digital signatures of X, Y, and T, respectively. Each signature is computed over the message fields preceding it (including the previous signatures) [3].

both the source and the target and a request id. Intermediate nodes that receive the request for the first time append their id to the node list and rebroadcast the request. A typical route request m_j that is broadcast by a node X_j, $0 \leq j \leq i$, on a route $S = X_0, X_1, ..., X_i, X_{i+1} = T$, is of the form

$$m_j = (rreq, S, T, id, (X_1, ..., X_j)).$$

When the target receives the route request, it creates a route reply that contains the ids of the source and the target, the node list found in the request, and a digital signature computed by the target over the elements preceding the signature. The reply is unicast upstream through the nodes in the request in the reverse order. When and intermediate node receives the reply, it performs the following verifications:

1. its id is in the accumulated route contained in the reply.

2. the id preceding it and the id next to it in the node list are its neighbors.

3. the signatures in the route reply are valid.

If these verifications succeed, the intermediate node signs the reply and forwards it to the next node in the node list towards the source. As presented in [3], a route reply unicast by X_j, $0 \leq j \leq i$, is of the form

$$m_j = (rrep, S, T, id, (X_1, ..., X_i), (sig_T, ..., sig_{X_j})),$$

where sig_{X_j} is the digital signature of X_j on the message fields preceding it.

When the source S receives the route reply, it checks if X_1 (i.e., the first id in the node list) is one of its neighbors. Then it verifies each individual signature in the reply. S accepts the route returned in the reply if all these verifications succeed. Figure 2.2 illustrates the operation of endairA.

Chapter 3 Greedy Routing Protocol with Backtracking for Mobile Ad-Hoc Networks

3.1 Introduction

As mentioned earlier, routing protocols designed for MANETs can be broadly classified as geographic routing protocols (or position-based routing protocols) and topology-based routing protocols. In geographic routing protocols, nodes do not maintain information related to network topology (i.e., they are topology independent). Generally [6], nodes need their own location, their neighbors' location, and the location of the destination node to which the packet needs to be forwarded. Using this location information, routing is accomplished by forwarding packets hop-by-hop until the destination node is reached [7]. Greedy forwarding (GPSR [8]), is one of the main strategies used in geographic routing protocols. Under Greedy forwarding, an intermediate node on the route forwards packets to the next neighbor node that is closer to the destination than itself. Different geographic routing protocols use different greedy forwarding strategies which can be defined in terms of distance, progress, and/or direction towards destination nodes.

Unlike on-demand routing protocols, geographic routing protocols do not depend on flooding route request messages to discover routes. This feature helps geographic routing protocols to reduce the extra overhead incurred for route discovery [2, 7] . A node only needs to know the position of its neighbors and the position of the destination to forward packets. Therefore, geographic routing protocols generally are more scalable than topology based routing protocols [18, 87, 17]. In spite of the benefits mentioned above, geographic routing protocols have the following limitations: Greedy forwarding, the primary packet forwarding strategy used by geographic routing protocols, may fail in low density networks, networks with non-uniformly distributed nodes, and/or networks where obstacles can be present.

3.1.1 Objective

From the geographic routing protocols we discussed above, we observe that the more robust the protocol, the less scalable it is. Furthermore, many of the existing geographic routing protocols either use more control information (to make it more Robust) which may result in redundant messages, contention, and collision, or use less control information (to make it more scalable) which may lead to less packet delivery ratio. In this chapter, we address this issue and propose Greedy Routing Protocol with Backtracking (GRB), a novel and simple position-based routing protocol which allows each node to forward data packets to its ***best neighbor*** (method used for best neighbor selection is explained in Section 3.2.1) possible until the destination is reached. Unlike GPSR, GRB uses less computation to determine the next hop on the route and it performs as well as or better than GPSR, AODV, and DSR.

The rest of the chapter is organized as follows. In Section 3.2, we present our protocol. In Section 3.3, we present the performance evaluation results of our protocol.

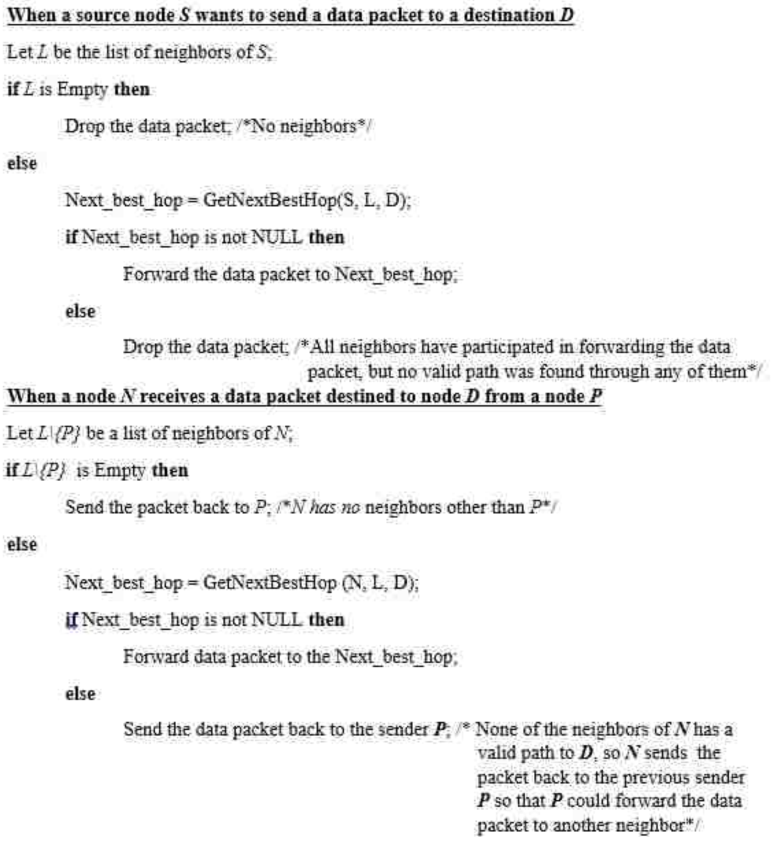
3.2 Our Greedy Routing Protocol with Backtracking (GRB)

In this section, we present the basic idea behind our protocol, and then present detailed description of the protocol.

3.2.1 Basic Idea Behind GRB

GRB routes data packets either in forwarding mode (greedy mode or simple forwarding) or in backtracking mode. When a sender/intermediate node S wants to send/forward a packet to a destination D, it picks the *best neighbor* $N1$ and sends the packet to $N1$. The best neighbor $N1$ is the neighbor that is closest to the destination than any other neighbor; note that this neighbor may not be closer to the destination than S itself because S may be facing a void. If the packet backtracks from this node to S, it picks the one that is closest to the destination among the remaining neighbors, and this process continues until all neighbors have been tried; if it cannot forward the packet through any of its neighbors, it sends the packet back to the node from which it received the packet. Every node on the path uses the same strategy to forward packets. Note that if the node picked is closer to the destination than S, then the forwarding is implicitly greedy; otherwise, the node tries to forward packets around a void. In this protocol, a source node drops data packets if it has no neighbors or it tried all the neighbors to forward the packet and failed.

Formal description of the protocol for data forwarding and finding the next best hop is given in Figures 3.1 and 3.2.



```
When a source node S wants to send a data packet to a destination D
Let L be the list of neighbors of S;
if L is Empty then
        Drop the data packet; /*No neighbors*/
else
        Next_best_hop = GetNextBestHop(S, L, D);
        if Next_best_hop is not NULL then
                Forward the data packet to Next_best_hop;
        else
                Drop the data packet; /*All neighbors have participated in forwarding the data
                                        packet, but no valid path was found through any of them*/
When a node N receives a data packet destined to node D from a node P
Let L\{P} be a list of neighbors of N;
if L\{P}  is Empty then
        Send the packet back to P; /*N has no neighbors other than P*/
else
        Next_best_hop = GetNextBestHop (N, L, D);
        if Next_best_hop is not NULL then
                Forward data packet to the Next_best_hop;
        else
                Send the data packet back to the sender P; /* None of the neighbors of N has a
                                                              valid path to D, so N sends  the
                                                              packet back to the previous sender
                                                              P so that P could forward the data
                                                              packet to another neighbor*/
```

Figure 3.1: GRB Data Forwarding (Sending/Receiving Data Packets).

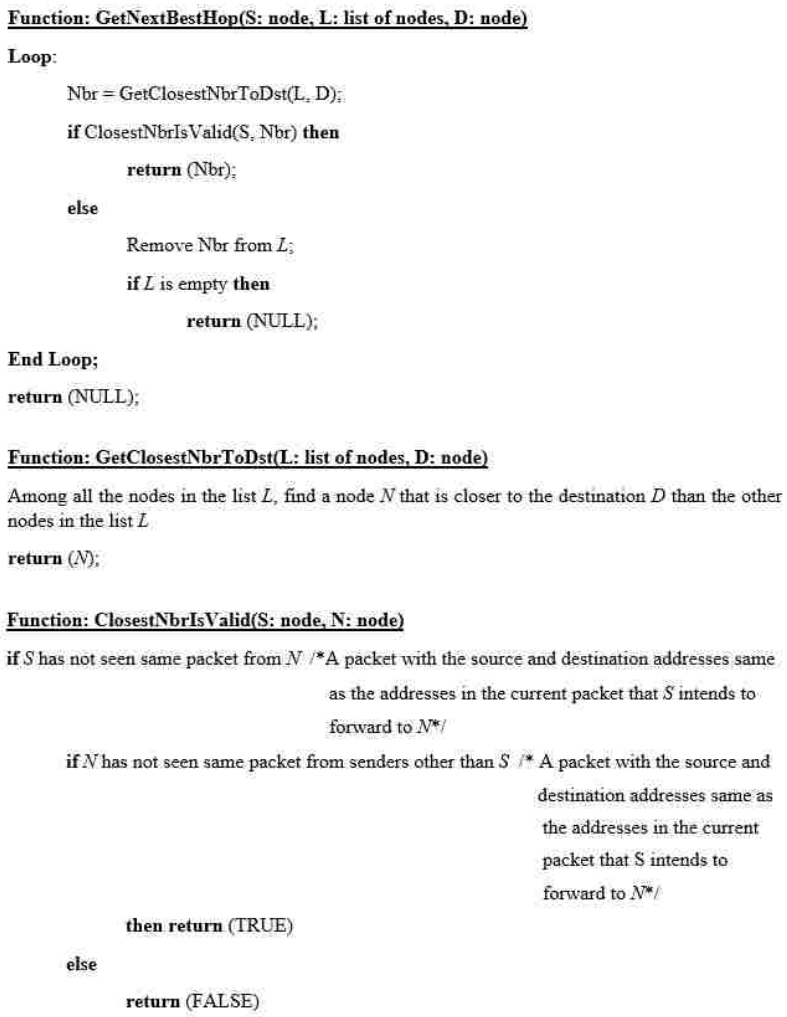
3.2.2 Assumptions

We assume that all nodes have the same transmission range (i.e., all links are bidirectional). We also assume that each node is equipped with a GPS and each node can get the location of the destination node through an available Location Service. In the following subsections we describe our protocol in detail.

3.2.3 Data Structures Used in the Protocol

Each node maintains the following two tables.

Neighbor Table. Each node sends a *HELLO* packet to all its neighbors in each



```
Function: GetNextBestHop(S: node, L: list of nodes, D: node)
Loop:
    Nbr = GetClosestNbrToDst(L, D);
    if ClosestNbrIsValid(S, Nbr) then
        return (Nbr);
    else
        Remove Nbr from L;
        if L is empty then
            return (NULL);
End Loop;
return (NULL);

Function: GetClosestNbrToDst(L: list of nodes, D: node)
Among all the nodes in the list L, find a node N that is closer to the destination D than the other
nodes in the list L
return (N);

Function: ClosestNbrIsValid(S: node, N: node)
if S has not seen same packet from N  /*A packet with the source and destination addresses same
                                        as the addresses in the current packet that S intends to
                                        forward to N*/
    if N has not seen same packet from senders other than S  /* A packet with the source and
                                                               destination addresses same as
                                                               the addresses in the current
                                                               packet that S intends to
                                                               forward to N*/
        then return (TRUE)
    else
        return (FALSE)
```

Figure 3.2: GRB Data Forwarding (Functions).

time interval T. This $HELLO$ packet includes the node's id as well as its position. To minimize collision of $HELLO$ packets due to concurrent transmissions, we jitter each $HELLO$ packet transmission interval by R milliseconds between two successive transmissions of $HELLO$ packets so that each node transmits $HELLO$ packets at a random time chosen in the interval $[T-R, T+R]$. When a node receives a $HELLO$ packet, it creates in its Neighbor Table an entry containing neighbor identifier (NbrID), neighbor position, and lifetime if that neighbor is not in the table; if there is an entry corresponding to that neighbor in the table, the lifetime is updated. If a node does not receive $HELLO$ packets for a time longer than $2T$ from a neighbor node, it assumes the neighbor has moved and removes the associated entry from the table.

Seen Table. This table helps picking ***best neighbor*** for forwarding packets to the destination. For that purpose, when a node receives a data packet, it stores the information about the packet in its Seen Table. As shown in Table 3.1, each record of this table contains five fields namely, neighbor ID (NbrID), source address (Src), destination address (Dst), flag (Flag), and lifetime (Lifetime). $NbrID$ is the address of the neighboring node that has sent the packet, forwarded the packet, or the node from which the packet has backtracked. Src contains the address of the source node that generated the data packet. $Flag$ indicates whether the received packet is a new packet (i.e., forwarding mode) or it has backtracked from a neighboring node (i.e., backtracking mode). The flag is set to $FALSE$ when the packet is in forwarding mode and set to $TRUE$ when it has backtracked. The lifetime field specifies the lifetime of the associated record in the Seen Table. When a node receives a data packet, it creates an entry in its Seen Table if the packet is new. However, if it has received a data packet with the same source and destination addresses from the same neighboring node, then it updates the lifetime of the associated record. On the other hand, when the lifetime expires, the associated record with that lifetime is removed from the table.

3.2.4 Sending and Forwarding Packets

Each node can send, forward, and/or receive data packets. When a node has data packets to send and the destination node is not one of its neighbors, it picks the best neighbor as described in Section 3.2.1 and forwards the packet to that neighbor. Since our protocol does not enforce the next-hop N to be closer to the destination than the sender S, N is either closer to D than S (i.e., Greedy mode), or farther to D than S. However, the next-hop N must be closer to D than any other neighbor that has not seen a packet to the same source-destination pair according to their Seen Table. Before forwarding the packet to N, the source or intermediate node S does the following:

S looks up its Seen Table for N. If S finds a record for N in the Seen Table that has the same source and destination addresses as that in the packet, then it considers N as invalid next hop for that packet and picks another neighboring node as the next hop. This means that S has received this packet from N which is either a new packet (i.e., flag is FALSE) or a backtracked packet (i.e., flag is TURE). It cannot forward the packet to that node because that results in a loop. For example, in Figure 3.3, when node $N3$ receives a data packet from node $N1$, it creates an entry in its Seen Table as shown in Table 3.1. This entry tells $N3$ that $N1$ is invalid next hop because it has received the packet from $N1$ and as a result, the Seen Table prevents loop between $N3$ and $N1$. However, the Seen Table of $N1$ does not have $N3$ in the table so it can forward the data packet to $N3$ after checking with $N3$ if it is a valid next hop. How

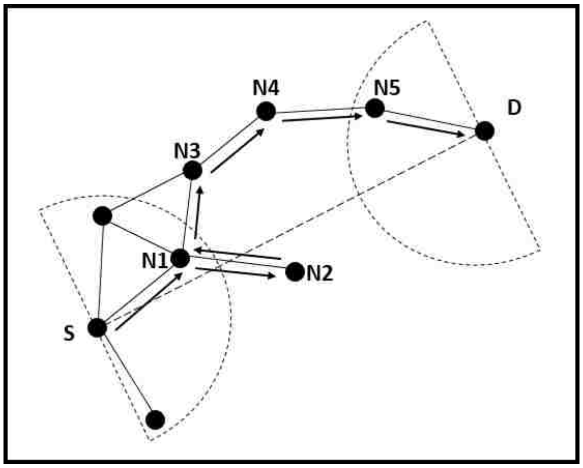


Figure 3.3: Data forwarding Example.

this verification is done is explained next.

Table 3.1: Seen Table at Node $N3$ in Figure 3.3

NbrID	Src	Dst	Flag	Lifetime
N1	S	D	False	T

How S verifies with N if it is a valid next hop. If N is not in the Seen Table of S, then S sends N a verification packet, with same source-destination pair in the header as in the data packet's header, asking N to check whether it has seen data packets from the same source-destination pair from any of its other neighbors. When N receives the verification packet, it checks its Seen Table for an entry that has the same source and destination addresses as that in the verification packet, with a Flag set to $False$, but with NbrID different from the ID of S. If such an entry is found, it means that N has seen a packet for the same source-destination pair and it sends a reply back to S indicating that it is invalid next hop. However, if such entry is found but Flag set to $True$, it means a neighbor $N1$ of node N has sent back the data packet to N after $N1$

failed to forward the packet. In this case, maybe there are neighbors of N other than $N1$ that have not been chosen by $N1$ to forward a packet yet, therefore N is not considered as invalid next hop and as a result, N sends a reply back to S indicating that it is a valid next hop for that data packet. After receiving the reply from N, if S finds N is a valid next hop, it forwards the packet to N. Otherwise, it picks another neighbor as a new candidate for next hop and checks if it is a valid next hop and so on. For example, in Figure 3.3, when $N1$ decides to forward a data packet to $N3$, it sends a verification packet to $N3$. $N3$ checks its Seen Table for an entry with NbrID set to any ID other than $N1$, same Src and Dst values as those in the verification packet, and Flag set to $False$. However, $N3$ does not have such entry in its Seen Table (refer to Table 3.1); hence $N3$ sends a positive reply to $N1$, and $N1$ forwards the packet to $N3$. If a node finds all its neighbors are invalid next hops, then the packet is sent back to the node from which it was received.

When a packet backtracks. A packet backtracks from the current node to its sender in the following two cases:

1. All the neighbors of the current node have seen that packet. This means none of the neighbors could forward the packet.
2. The current node has no neighbors other than the sender. For example, in Figure 3.3, $N2$ has no neighbors other than $N1$ which sent the packet to it. Therefore, the packet backtracks to $N1$ and $N1$ inserts a new entry to its Seen Table as shown in Table 3.2. The Flag of the new entry (i.e., second

row) is set to $True$ which means that from the perspective of $N1$, $N2$ is considered invalid next hop for that packet. Therefore, when $N1$ tries to pick next hop for the same destination next time, it will not pick $N2$ as long as the Lifetime of the associated entry (i.e., second row in Table 3.2) in the Seen Table of $N1$ has not expired.

Table 3.2: Seen Table at Node $N1$ in Figure 3.3

NbrID	**Src**	**Dst**	**Flag**	**Lifetime**
S	S	D	False	T1
N2	S	D	True	T2

When a packet is dropped. A packet is dropped by a node when all the neighbors have been identified as invalid next hops or the node has no neighbors.

3.3 Performance Analysis

In this section, we present the performance evaluation results of GRB compared to AODV [15], DSR [33], and GPSR [8]. We first describe the simulation environment and then discuss the simulation results. We simulated GRB, AODV, and DSR on a variety of network topologies. Then we compared the performace of GRB with the results provided in GPSR [8].

3.3.1 Simulation Environment

We used GloMoSim [88], a network-simulation tool for studying the performance of routing protocols for MANETs, for evaluating the performance of GRB. We chose IEEE 802.11 and IP as the MAC and network layer protocols, respectively. All nodes have a fixed transmission range of 250 m. We used the implementation of AODV and DSR that comes with the GloMoSim 2.0.3 package to compare their performance with GRB. We ran several simulations on two different sets of traffic flows. The simulations run in different terrain areas are given in Table 3.3; each simulation lasted for 900 seconds of simulated time. The nodes were distributed uniformly at random in the terrain area. We used the following four metrics to evaluate performance:

1. Packet Delivery Ratio: Measures the success rate of delivered data packets.
2. End-To-End Delay: Average time a packet takes to reach the destination node.
3. Hop Count: The average number of hops a packet traverses to reach the destination.
4. Node Density: Number of nodes in the area.
5. Network Diameter: For studying the effect of different network areas on the performance of the protocol.

In this experiment, we varied the number of nodes simulated from 50 to 300. Two sets of random traffic flows have been used in the simulation. The first set consisted of 30 CBR (Constant Bit Rate) flows in which 30 different senders generate data packets to be sent to 30 random destinations. Each CBR flow sends packets at speed of 16Kbps and uses 512-byte packets. Depending on the start time and end time of each sender in each flow, different number of packets are sent by different CBR flows. However, in each flow, each sender sends a packet every 0.25 second. Node mobility is set using random Way-point [33] model. Under this model, each node travels from a location to a random destination at a random speed, the speed being uniformly distributed in a predefined range. After a node reaches its destination, it pauses for a predetermined amount of time and then moves to a new destination at a different randomly chosen speed. In our simulation, the speed randomly chosen lies between 0 and 20 meters/second. In order to study how mobility affects the performance of the routing protocols, we selected pause times of 0, 20, 30, 40, 60, 80, 100, and 120 seconds. When the pause time is 0 seconds, every node moves continuously. As the pause time increases, the network approaches the characteristics of a fixed network. The second set consists of 20 CBR flows which has 20 sender nodes generating packets at a speed and size same as that in the first set.

Table 3.3: Topology used for Simulation

Nodes	Network Area	CBR Flows	Packets Sent
{50,75,100,125,150}	1500m X 1500m	30	8780
{175,200,225,250,300}	1500m X 1500m	30	8780
50	1500m X 300m	30	8780
112	2250m X 450m	30	8780
200	3000m X 600m	30	8780

3.3.2 Packet Delivery Ratio

The overall average packet delivery ratio for DSR, AODV, and GRB are 55.52%, 97.38% and 98.60%, respectively. We selected CBR flows randomly; hence it is not known whether there is a valid path between the source node and the destination node in each flow. Higher number of packets (refer to Table 3.3) imposes higher demand on routing protocols as higher traffic is generated between source and destination pairs. GRB finds next hops based on the most up to date location information of the nodes involved in the forwarding process. It simply picks next hops based on Seen Tables to forward data packets which results in few control packets. This makes GRB adapt to location changes; hence it tolerates mobility better than AODV and DSR. Therefore, GRB delivers higher number of data packets than DSR and AODV for different pause times as shown in Figures 3.4, 3.5, 3.6, and 3.7.

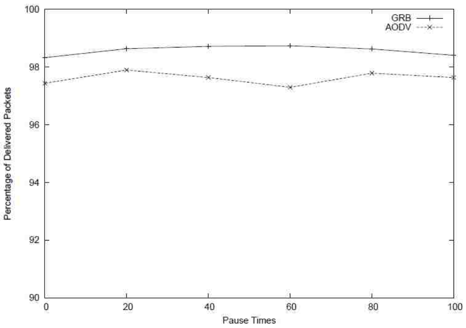


Figure 3.4: Packet Delivery Ratio (50 Nodes, 30-CBR Flows, network area (1500m x 300m)), GRB compared with AODV.

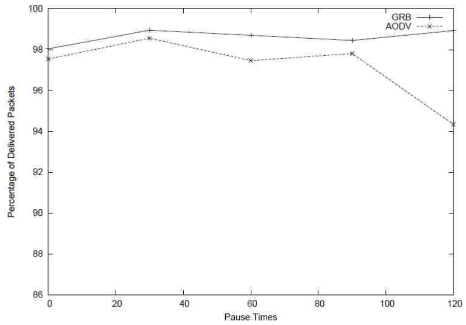


Figure 3.5: Packet Delivery Ratio (50 Nodes, 20-CBR Flows, network area (1500m x 300m)), GRB compared with AODV.

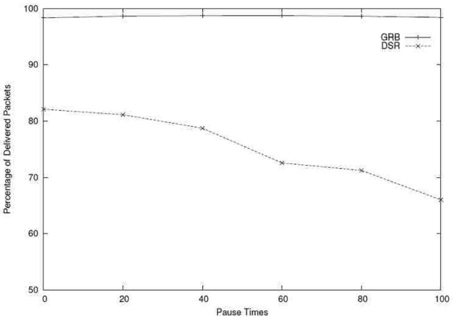


Figure 3.6: Packet Delivery Ratio (50 Nodes, 30-CBR Flows, network area (1500m x 300m)), GRB compared with DSR.

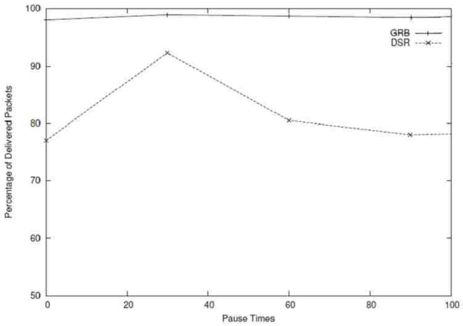


Figure 3.7: Packet Delivery Ratio (50 Nodes, 20-CBR Flows, network area (1500m x 300m)), GRB compared with DSR.

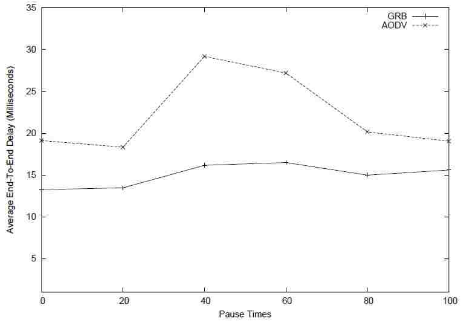


Figure 3.8: Average End-To-End Delay (50 Nodes, 30-CBR Flows, network area (1500m x 300m)), GRB compared with AODV.

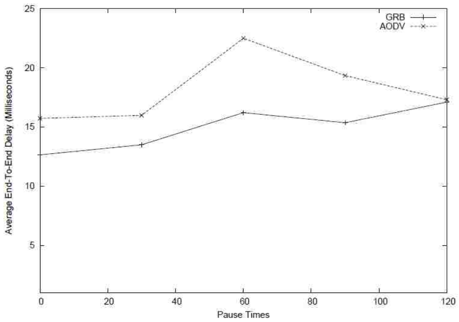


Figure 3.9: Average End-To-End Delay (50 Nodes, 20-CBR Flows, network area (1500m x 300m)), GRB compared with AODV.

3.3.3 End-To-End Delay

As shown in Figures 3.8 and 3.9, the overall average end-to-end delay for AODV and GRB are 22.17 milliseconds and 14.98 milliseconds, respectively. For each CBR flow, we take the average end-to-end delay of all the packets received by the destination node in that flow. Then, we take the average delay of all the CBR flows. Because of its simplicity, GRB takes less time to deliver data packets in most of the scenarios. As shown in Figures 3.8 and 3.9, packets take more than 18 milliseconds on average to reach their destinations under AODV whereas GRB delivers packets in less than 16 milliseconds. We can see GRB delivers packets much faster when network size and area is moderately small (50 nodes, (1500m x 300m) area). That is because most of the packets find greedy paths which take less calculation time and the decision is taken quickly based on the neighbors' location information and their status regarding whether or not they are valid next hops. However, under AODV, data packets should wait for the route to be set up. Moreover, routes discovered under AODV may not be shorter than those discovered under GRB, because GRB uses greedy approach.

As a result, AODV results in higher average end-to-end delay.

3.3.4 Node Density

Since our protocol uses only information about neighbors in forwarding decision, as node density increases, GRB keeps delivering higher fraction of data packets than AODV and DSR as shown in Figures 3.10 and 3.11. That is because both AODV and DSR depend on end-to-end route to forward data packets and that route is affected by mobility of the nodes and the size of the network. Therefore, due to mobility, more frequent link breaks occur leading to more route repair and setup and as a result, packets are lost more frequently under AODV and DSR. However, since the average end-to-end delay is taken only for packets that are delivered to their destinations and because data packets follow existing routes which decreases waiting time, AODV routes data packets in slightly less time than GRB when number of nodes grows to more than 200 as shown in Figure 3.13. However, GRB is faster in smaller networks (less density) because less computation required by nodes to make forwarding decisions since nodes have less neighbors. Average hop count is another parameter that we measure in this simulation to show that our protocol routes data packets with less number of hops as node density increases. For this metric, only the successfully delivered data packets are counted in the simulation results for both GRB and AODV. As shown in Figure 3.12, in smaller networks (i.e., less than 150 nodes), AODV uses less number of hops to forward data packets than GRB because there are more voids in sparse networks. This makes GRB data packets to go around voids through either next best hop or backtracking technique which makes GRB packets traverse more hops than AODV. However, as number of nodes increases, number of voids decreases and data packets move through greedy paths; hence GRB uses less number of hops than AODV in dense networks. It is worth mentioning that under GRB, average hop count is also reduced because next hop is chosen greedily.

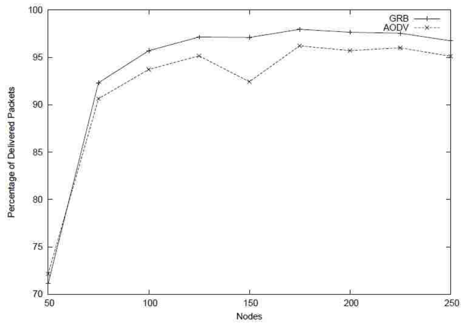


Figure 3.10: Packet Delivery Ratio as Number of Nodes increases (Network Area (1500x1500)), GRB compared with AODV.

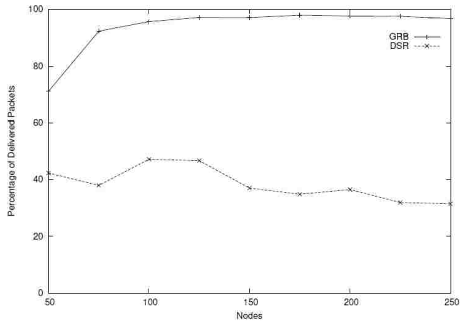


Figure 3.11: Packet Delivery Ratio as Number of Nodes increases (Network Area (1500x1500)), GRB compared with DSR.

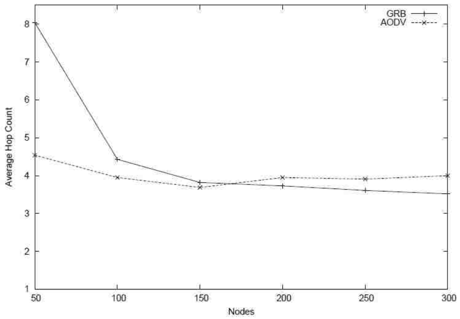


Figure 3.12: Average Hop Count as Number of Nodes increases (Network Area (1500x1500)), GRB compared with AODV.

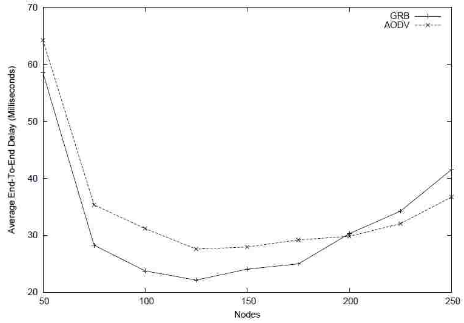


Figure 3.13: Average End-To-End as Number of Nodes increases (Network Area (1500x1500)), GRB compared with AODV.

3.3.5 Network Diameter

Figures 3.14 and 3.15 present packet delivery ratio results and Figures 3.18 and 3.19 present average hop count results for 112-nodes and 200-nodes networks with same

CBR traffic and same node density for both networks. In these simulations, the terrain area within which nodes move are (2250x450) meters and (3000x600) meters respectively. We also evaluated the effect of changing network diameter on success rate and hop count for both GRB and AODV. The probability of an established route breaking increases as the routes grow longer. GRB delivers higher percentage of packets than AODV and DSR at all pause times on larger networks because GRB incurs no penalty as the path length increases from source nodes to destination nodes. Moreover, GRB recovers from loss of neighbor (next hop) instantaneously by simply finding another *candidate next hop* which will take over the forwarding process. However, percentage of packets delivered under AODV decreases considerably as the network diameter increases because it needs to maintain longer end-to-end routes. DSR incurs higher traffic overhead in wider networks since it needs to maintain longer end-to-end source routes; hence its success rate decreases accordingly and it is much lower than that of GRB as shown in Figures 3.16 and 3.17. For the hop count metric, we calculate the average of all the received packets by all the destination nodes in all the flows for both GRB and AODV routing protocols. In small areas, AODV has less hop counts in higher mobility rates (i.e., lower pause times); however, GRB uses less hop counts when node mobility decreases (i.e., higher pause times) because the Seen Table entries will be more accurate as nodes remain for longer time in their current locations. This gives GRB better chance to direct data packets through valid paths. In wider networks (i.e., larger diameter), GRB uses less hop counts than AODV for all the pause times (i.e., for low and high mobility rates) because in such networks, routes established under AODV break more often due to longer end-to-end routes. Since same node density is used for both networks, it does not cost GRB any additional calculation since forwarding decisions are made locally, so it remains using less hops than AODV.

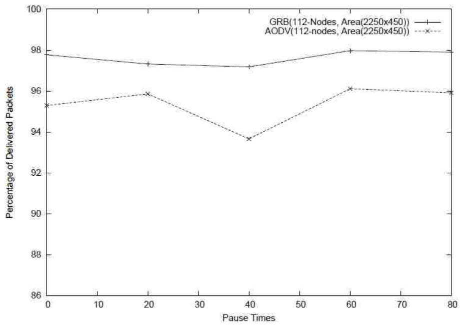


Figure 3.14: Packet Delivery Ratio of Network Area (2250x450), 112 nodes, 30-CBR Flows, GRB compared with AODV.

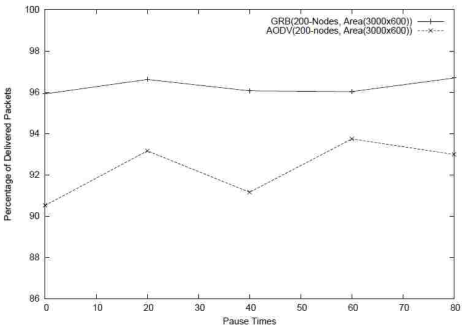


Figure 3.15: Packet Delivery Ratio of Network Area (3000x600), 200 nodes, 30-CBR Flows, GRB compared with AODV.

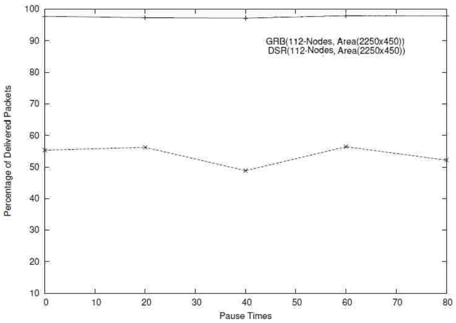


Figure 3.16: Packet Delivery Ratio of Network Area (2250x450), 112 nodes, 30-CBR Flows, GRB compared with DSR.

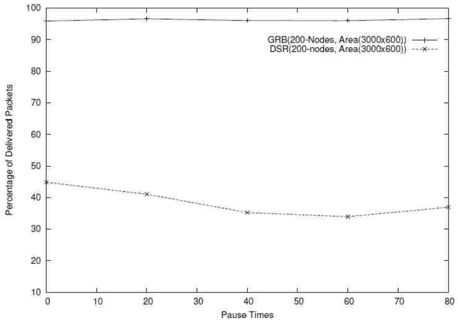


Figure 3.17: Packet Delivery Ratio of Network Area (3000x600), 200 nodes, 30-CBR Flows, GRB compared with DSR.

3.3.6 GRB Vs. GPSR

Even though we didn't simulate GPSR, we used the performance results published for GPSR in [8] and the results we obtained for GRB to compare the performance of the two protocols. As stated in [8], GPSR performance evaluation counts only those

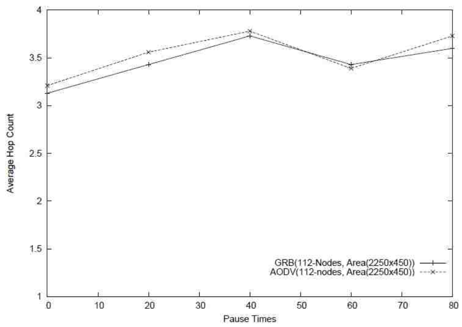


Figure 3.18: Average Hop Count of Network Area (2250x450), 112 nodes, 30-CBR Flows, GRB compared with AODV.

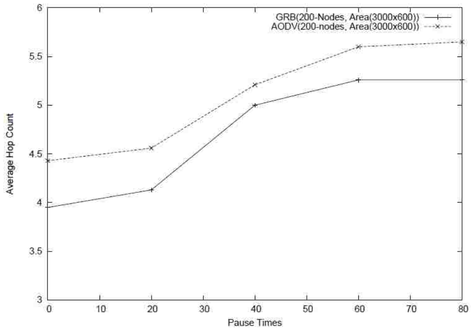


Figure 3.19: Average Hop Count of Network Area (3000x600), 200 nodes, 30-CBR Flows, GRB compared with AODV.

packets for which a path exists to the destination. We used the same input settings as those used for GPSR to compare the success rate. The settings are: 50 nodes, 30 CBR flows, pause times (PT) (0, 30, 60, and 120) seconds, area (1500x300) meters, and node density ($1node/9000m^2$). From the results presented in GPSR [8], successful packet delivery rate (SPDR) of GPSR ranges from 95% to 99.10% for pause time 0 second, while GRB successfully delivers 98.93% of the total packets sent. When pause time is 30 seconds, GPSR achieves success rates between 98.20% to 99.70% whereas GRB achieves a success rate of 99.33%. When pause time is 60 seconds, GPSR delivers from 98.70% to 99.40% of data packets sent, while GRB delivers 99.04% of all packets. Finally, when pause time is 120 seconds, GPSR's packet delivery rate ranges from 98.60% to 99.40% and GRB's packet delivery rate is 99.28%. The results of the comparison is shown in Table 3.4.

As shown in Figure 3.20, for 50 and 112 nodes, pause times 0 and 60 seconds, areas (1500x300) and (2250x450) meters, we see that GRB performs better than GPSR with respect to Successful Packet Delivery Ratio (SPDR). We noticed that the performance

Table 3.4: Input Settings and Corresponding Results for both GRB and GPSR

Nodes	**Network Area**	**PT(s)**	**SPDR(GRB)**	**SPDR(GPSR)**
50	1500m X 300m	0	98.98	97.04
50	1500m X 300m	60	99.07	98.16
112	2250m X 450m	0	98.00	97.50
112	2250m X 450m	60	98.54	98.25
200	3000m X 600m	0	97.02	95.00
200	3000m X 600m	60	96.84	97.50

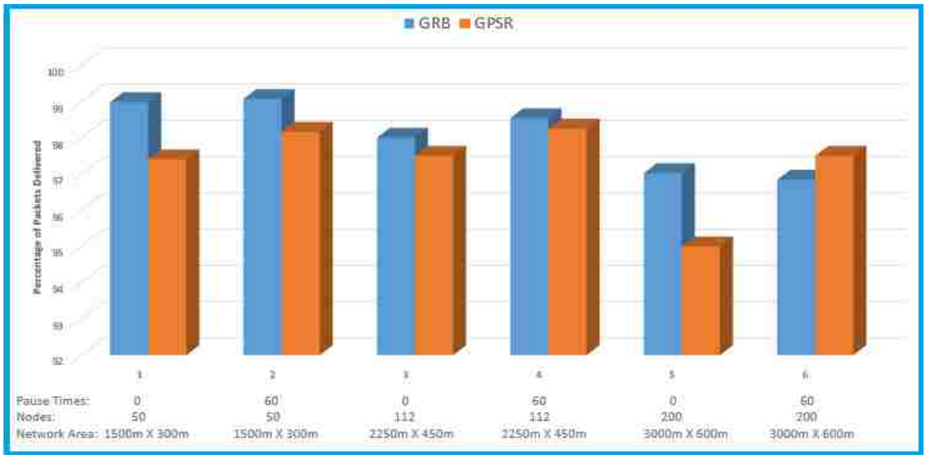


Figure 3.20: GRB Vs. GPSR.

of our protocol in successfully delivering data packets is almost same as that of GPSR in some cases and higher than GPSR in other cases. However, when greedy routing fails due to a void in the direction of the destination, GPSR has to planarize the local network graph and use it to route around voids. Planarizing the graph results in computation overhead as well as routing failure. Figures 3.21, 3.22, and 3.23 show how GRB successfully routes around voids where GPSR could not because of the planarization and face routing problems discussed in Section 1.6. When face routing fails because of unidirectional links as shown in Figure 1.4, under GRB, the first packet follows the links labeled (1,2,3,4,5,6,7,8,9,10,11) to reach the destination as shown in Figure 3.21. However, the remaining packets will be routed via the links

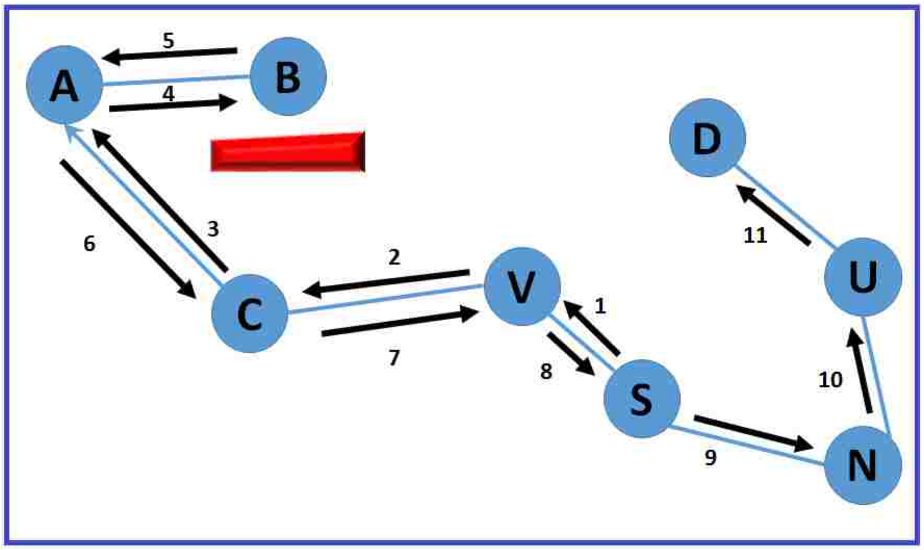


Figure 3.21: GRB Succeeds when Unidirectional Links Cause Routing Failure.

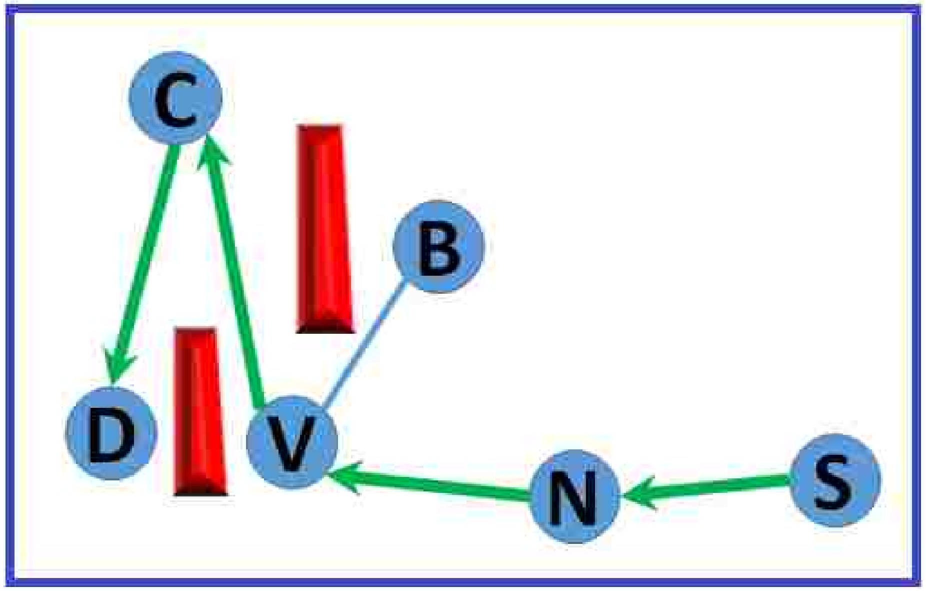


Figure 3.22: GRB Succeeds when Disconnected Links Cause Routing Failure.

labeled (9,10,11) because the first packet backtracked from V to S. Hence, in the *Seen Table* of the source S, it is stated that node V is an invalid next hop for the destination D.

When planarization of the local network graph results in disconnected links as shown in Figure 1.5, data packets under GRB follow the sold arrows (i.e. nodes

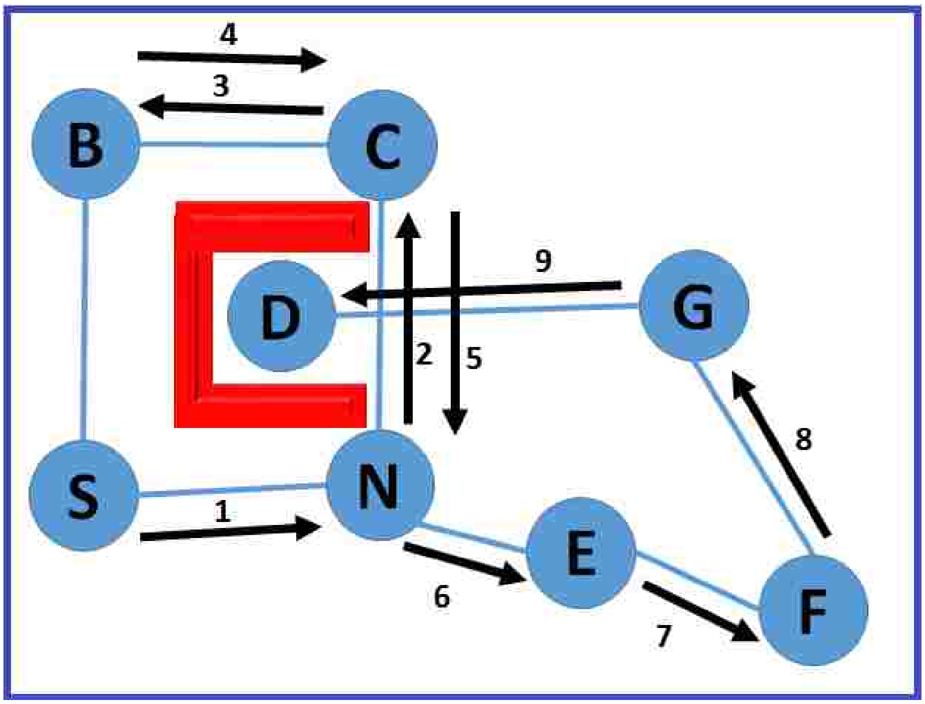


Figure 3.23: GRB Succeeds when Cross Links Cause Routing Failure.

(N, V, C)) to reach the destination D as shown in Figure 3.22. When cross links caused data packets to loop as shown in Figure 1.6, the first packet follows the links labeled (1,2,3,4,5,6,7,8,9) under GRB as shown in Figure 3.23. Moreover, the remaining packets follow the path labeled (6,7,8,9) towards D.

GRB neither requires planarizarion of local network graph nor it switches from greedy mode to an alternative mode when a packet faces void; instead a node simply selects the *best next hop* without imposing the condition that the next hop be closer to the destination than itself. GRB depends on the *Seen Table* to determine the next hop. Therefore, GRB needs less control information which makes it better and robust. So, we can see the difference between GRB and GPSR in delivering data packets is around 0.0066% in one scenario keeping in mind that GRB outperforms GPSR in some other scenarios. When compared to the simplicity of GRB and the less control overhead GRB needs, this difference is negligible. It is worthy to recall that we select CBR flows randomly without knowing whether there exists a route corresponding to each flow. However, under GPSR, according to the authors: “Only packets for which a path exists to the destination are included in the graph”.

Chapter 4 Hybrid On-demand Greedy Routing Protocol with Backtracking for Mobile Ad-Hoc Networks

4.1 Introduction

Topology-based routing protocols which depend on the current topology of the network can be classified into three main types, namely, proactive routing protocols, reactive routing protocols (i.e., on-demand routing protocols), and hybrid routing protocols [16, 89, 19]. Under proactive routing protocols (e.g., DSDV [13]), nodes establish table-based routes [14] to all other nodes. This makes routes reliable and nodes don't wait for route discovery which decreases latency. On the other hand, overhead incurred for maintaining up to date routes at each node limits scalability. In addition to that, consume lot of memory especially when network size grows.

In on-demand routing protocols, senders build and maintain routes to destination nodes only when they need it. On-demand routing protocols need less memory and storage capacity than proactive routing protocols. However, in networks where nodes are highly mobile, route discovery may fail since the path can be long and links may break due to node mobility while establishing routes [16]. Routes can also break during packet transmission due to mobility of nodes on established routes. The delay caused by route discovery for each data traffic can increase latency (i.e., end-to-end delay).

4.1.1 Objectives

Many of the on-demand routing protocols such as DSR [33], and AODV [15] flood route requests throughout the network which results in high control overhead due to redundant propagation of route requests. On the other hand, geographic routing protocols such as GPSR [8] and its variations do not require the establishment of routes to the destination a priori for sending packets. Under geographic routing protocols such as GPSR, each node receiving a data packet forwards the packet to one of its neighbors that is closer to the destination than itself; different protocols use different strategies for picking such a neighbor. If a node does not have such a neighbor (we say there is a void in the direction of the destination), then it constructs the graph of the local network and tries to route around the void using that graph. This incurs additional overhead due to the construction of the graph. We address these disadvantages of on-demand and geographic routing protocols and propose a novel on-demand routing protocol, called Hybrid Greedy On Demand Routing Protocol with Backtracking (HGRB). HGRB uses geographic approach for forwarding route requests during route discovery. If the route request (RREQ) packet meets a void, it uses a simple back-tracking approach to forward RREQs around the voids. Performance evaluation shows that HGRB significantly outperforms both AODV and DSR.

In this chapter, we present a novel on-demand routing protocol which uses position-based greedy forwarding with back-tracking for route discovery. Our protocol performs better than some of the well-known on-demand routing protocols with respect to control overhead, packet delivery ratio and the number of hops on the routes discovered.

The rest of this chapter is organized as follows. Section 4.2 presents the proposed protocol (i.e., HGRB). In Section 4.3, we present the performance evaluation results of our protocol.

4.2 The Proposed Protocol (HGRB)

In this section, we present a detailed description of our protocol.

4.2.1 Assumptions

We assume that all nodes have the same transmission range (i.e., all links are bidirectional). We also assume that each node is equipped with a GPS and each node can get the location of the destination node through an available Location Service. We describe the basic idea behind HGRB next.

4.2.2 Basic Idea Behind HGRB

Under HGRB, when a source S needs to find a route to destination D, it picks the *best neighbor* $N1$ and sends the route request (RREQ) packet to $N1$. The best neighbor $N1$ is determined as follows: S picks the neighbor that is closer to the destination than all of its other neighbors. Note that this neighbor may not be closer to the destination than S itself because S may be facing a void in the direction of the destination. If the RREQ packet backtracks from $N1$ to S, S picks the one that is closest to the destination among the remaining neighbors, and this process continues until all neighbors have been tried; if it cannot forward the RREQ packet through any of its neighbors, it simply concludes that there is no route to the destination. When an intermediate node I receives the RREQ packet from a node N, it uses the same strategy to pick a neighbor to forward the packet; if it fails to forward the RREQ packet through any of its neighbors, it sends the packet back to N (i.e., the packet backtracks). This process continues until the RREQ packet reaches the destination or it has reached a pre-determined number of back-tracking. Section 4.2.3 explains in detail how the next hop is selected for forwarding RREQs. Note that if the next hop picked by a node for forwarding RREQ is closer to the destination than the node itself, then forwarding is implicitly greedy; otherwise, the RREQ packet is forwarded around the void using simple backtracking.

In case of backtracking, the nodes traversed backwards do not become part of the route because they are considered invalid next hops. For example, in Figure 4.1, when node S decides to discover a route to destination node D the RREQ packet first takes the path *(N1, N2, N3)*; However, $N3$ is invalid next hop. Therefore, the RREQ backtracks to $N1$ from where $N1$ chooses $N4$ as a new next hop. Therefore, the route from S to D discovered under HGRB will be *(N1,N4,N5,N6)*. In this case, the nodes traversed backwards (i.e., *N2* and *N3*) are not part of the established route. This helps in reducing the average hop count the data packets traverse.

Once the route is built between the source node and the destination node, data packets are transmitted from the source node to the destination node via that route. Under HGRB, a source node drops data packets if it has no neighbors or if it tried to forward the RREQ packet through all its neighbors and failed.

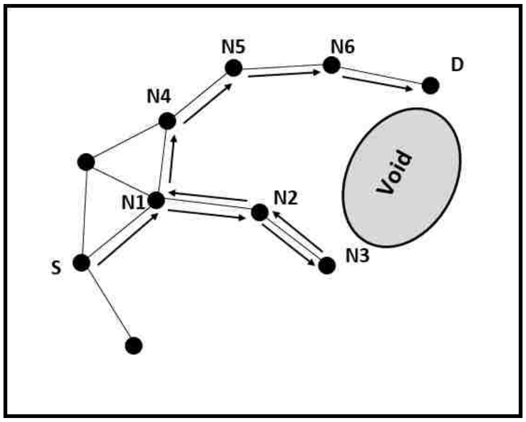


Figure 4.1: Route Setup Example.

4.2.3 Data Structures Used in HGRB

Nodes that participate in route discovery process maintain a *Seen Table* and a *Route Table* which are used during route discovery and data forwarding. Each node also maintains a *Neighbor Table*. How these tables are maintained and used is explained next.

- **Neighbor Table.** This table contains the location information of its neighboring nodes. Each node sends a $HELLO$ packet to all its neighbors in each time interval T. This $HELLO$ packet includes the node's id as well as its position. To minimize collision of $HELLO$ packets due to concurrent transmissions, we jitter each $HELLO$ packet transmission interval by R milliseconds between two successive transmissions of $HELLO$ packets so that each node transmits $HELLO$ packets at a random time chosen in the interval $[T - R, T + R]$. When a node receives a $HELLO$ packet, it creates in its Neighbor Table an entry containing neighbor identifier (NbrID), neighbor position, and lifetime of the neighbor. The lifetime of an entry in this table is updated whenever the node receives any packet(RREQ, data, route reply packet, etc.) from the neighbor associated with that entry.

- **Seen Table.** This table helps picking ***best neighbor*** for forwarding RREQ packets to the destination. For that purpose, when a node receives a RREQ packet, it stores the information about the packet in its Seen Table. As shown in Table 4.1, each record of this table contains five fields, namely, neighbor ID (NbrID), source address (Src), destination address (Dst), flag (Flag), and lifetime (Lifetime). $NbrID$ is the address of the neighboring node that has sent the RREQ packet, forwarded the RREQ packet, or the node from which the RREQ packet has backtracked. Src contains the address of the source node that generated the RREQ packet. $Flag$ indicates whether the received RREQ packet is a new packet (i.e., forwarding mode) or it has backtracked from a neighboring node (i.e., backtracking mode). This flag is set to $FALSE$ when the RREQ packet is in forwarding mode and set to $TRUE$ when it has backtracked. The lifetime field specifies the lifetime of the associated record. When a node receives a RREQ packet, it creates a record in its Seen Table for the neighbor from which the RREQ packet was received. A created record is removed from the table When its lifetime expires.

- **Routing Table.** This table has five fields namely, *Destination-Address*, the node to which the data packet, RREQ, or RREP is forwarded; *Lifetime*, the age of the associated record in the routing table; *Active*, a Boolean field which indicates whether the associated entry is active or not; if this field is set to

True, then the related record can be used for forwarding either control or data packets; *Source* indicates the source node that initiated the RREQ or a node that initiated a route reply (i.e., either the destination or an intermediate node that has an active route to the destination node). *Sequence-Number*, an integer field associated with each RREQ. Details about setting up routing tables are given in Section 4.2.4.

Best Next-Hop Selection and Verification

For selecting the next-hop N to forward a RREQ packet, the source or an intermediate node S does the following. It picks the neighbor N that is closer to the destination than any of the other neighbors that have not been considered for the next-hop selection and does the following verifications.

- **S looks up its Seen Table for N.** If S has an entry for N with the same associated source and destination addresses as that in the RREQ packet, then it considers N as an invalid next hop for that packet and picks another neighboring node as the next hop. This means that S has received this request from N which is either a new request (i.e., flag is FALSE) or a backtracked request (i.e., flag is TRUE), therefore, it cannot forward the RREQ packet to that node because that would result in a loop. For example, in Figure 4.2, when node $N3$ receives a RREQ from node $N1$, it creates an entry in its Seen Table as shown in Table 4.1. This entry tells $N3$ that $N1$ is an invalid next hop because it has received the same RREQ from $N1$ and as a result, the Seen Table prevents loop to occur between $N3$ and $N1$. However, the Seen Table of $N1$ does not have $N3$ as a neighbor node so it can forward the RREQ packet to $N3$.

- S **verifies with** N **if it is a valid next hop.** If N is not in the Seen Table of S, then S sends N a verification packet, with the same source-destination pair in the header as in the RREQ packet's header, asking N to check whether it has seen RREQ packets with the same source-destination pair from any of its other neighbors. When N receives the verification packet, it checks its Seen Table for an entry that has the same source and destination addresses as those in the verification packet, with a Flag set to $False$, but with a NbrID different from the ID of S. If such an entry is found, it means that N has seen a RREQ packet for the same source-destination pair from one of its other neighbors, and it sends a reply back to S indicating that it is an invalid next hop. However, if such an entry is found but the Flag is set to $True$, it means a neighbor $N1$ of node N has sent the RREQ packet back to N after $N1$ failed to forward the RREQ packet. In this case, there may be neighbors of N other than $N1$ that were not checked to forward the RREQ packet yet, therefore N is not considered as an invalid next hop and as a result, N sends a reply back to S indicating that it is a valid next hop for that RREQ packet. After S finds N to be a valid next hop, it forwards the RREQ packet to N. Otherwise, it picks another neighbor as a new candidate for next hop and checks if it is a valid next hop and so on. For example, in Figure 4.2, when $N1$ needs to send a RREQ packet to $N3$, it sends a verification packet to $N3$. $N3$ checks its Seen

Table for an entry with the NbrID set to any ID other than $N1$, with same Src and Dst values as those in the verification packet, and Flag is set to $False$. Since $N3$ does not have such an entry in its Seen Table (refer to Table 4.1), it sends a positive reply (i.e., $N3$ is a valid next hop) to $N1$, and $N1$ forwards the RREQ packet to $N3$. This verification process is necessary to prevent loops. For example, when node $N8$ receives the RREQ back from node $N9$, after it has verified with $N2$, $N8$ sends a verification packet to $N1$. $N1$ finds that it has seen a request (i.e., an entry) with a NbrID set to S which is different from $N2$, with same Src and Dst values as those in the verification packet, and the Flag set to $False$ (see Table 4.2). Therefore, $N1$ sends a negative response to $N8$ indicating that $N1$ is an invalid next hop for that request. Therefore, $N8$ sends the request back to $N2$ which in turn sends the request back to $N1$. If a node finds all its neighbors are invalid next hops, then the RREQ packet is sent back to the node from which it was received.

- **RREQ Packet backtracking.** A RREQ packet backtracks from the current node to its sender in the following two cases:

  1. All the neighbors of the current node have seen that packet. This means none of the neighbors could forward the packet.
  2. The current node has no neighbors other than the sender. For example, in Figure 4.2, $N9$ has no neighbors other than $N8$ which sent the request packet to it. Therefore, the packet backtracks to $N8$ and $N8$ inserts a new entry into its Seen Table as shown in Table 4.3. The Flag of the new

entry (i.e., second row) is set to $True$ which means that from the perspective of $N8$, $N9$ is considered an invalid next hop for that RREQ packet. Therefore, when $N8$ tries to pick the next hop for the same destination next time, it will not pick $N9$ if the Lifetime of the associated entry (i.e., second row in Table 4.3) in the Seen Table of $N8$ has not expired.

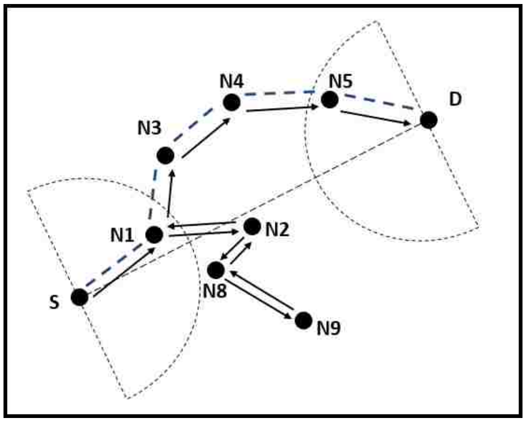


Figure 4.2: Route Setup Example.

Table 4.1: Seen Table at Node $N3$ in Figure 4.2

NbrID	Src	Dst	Flag	Lifetime
N1	S	D	False	T

Table 4.2: Seen Table at Node $N1$ in Figure 4.2

NbrID	Src	Dst	Flag	Lifetime
S	S	D	False	T1
N2	S	D	True	T2

Table 4.3: Seen Table at Node $N8$ in Figure 4.2

NbrID	Src	Dst	Flag	Lifetime
N2	S	D	False	T1
N9	S	D	True	T2

Table 4.4: Routing Table at Node $N8$ in Figure 4.2

Dst-Addr	**Next-hop**	**Lifetime**	**Active**	**Src**	**Sequence-Number**
S	N2	T	False	False	Seq

Table 4.5: Routing Table at Node $N5$ in Figure 4.2

Dst-Addr	**Next-hop**	**Lifetime**	**Active**	**Src**	**Sequence-Number**
S	N4	T	True	False	Seq
D	D	T	True	False	Seq

4.2.4 Route Discovery and Maintenance

In many of the existing on-demand routing protocols, when a source node wants to find a route to a destination, it broadcasts a RREQ message to all its neighbors. Every node that receives the RREQ message rebroadcasts the RREQ to all its neighbors. This method results in flooding broadcasts and incurs large overhead. In order to minimize such overlapped broadcasts, HGRB unicasts RREQ as follows: when a node needs to forward a RREQ packet, it picks the best neighbor N as described in Section 4.2.3 and sends the RREQ packet to that neighbor. When N receives the RREQ packet, it looks up its routing table to see whether it has an unexpired route to the destination node. If so, it sends a route reply (RREP) back to the source node. Otherwise, it forwards the RREQ to its next best neighbor. Since HGRB does not enforce the next-hop N to be closer to the destination than the sender S, N is either closer to D than S (i.e., Greedy mode), or farther away from D than S. However, the next-hop N must be closer to D than any other neighbor that has not seen a RREQ packet to the same source-destination pair.

Reverse Path Setup

When the RREQ propagates from a source node to its destination, it builds the reverse route from each node in the path back to the source node. When a node receives a RREQ from a neighbor node, it records the address of that neighbor, establishing the next hop on the route to the source node. If the receiving node is neither the desired destination nor it has an active route to the destination node, it forwards the RREQ to the next best hop as explained in Section 4.2.3. For example, when node $N8$ in Figure 4.2 receives a RREQ from node $N2$, it creates an entry in its routing table as shown in Table 4.4, and forwards the RREQ to the next hop $N9$. However, since $N9$ has no neighbors other than $N8$, $N9$ sends the RREQ back to $N8$. Even though it receives a RREQ from $N9$, $N8$ does not create a reverse path for that request because it is a RREQ packet that backtracked. The reverse path entries are maintained for a lifetime long enough for the RREQ to reach the destination and produce a reply to the source node.

Route Setup

When a RREQ arrives at the destination node or an intermediate node that has a valid route for the desired destination node, the receiving node N initiates a route reply. N unicasts a RREP packet back to the node from which it received the RREQ. By the time the RREQ arrives at N, a reverse path (see Section 4.2.4) from N to the source node (i.e., the node that initiated the RREQ) has been established. As the RREP propagates to the source node along the path traversed by the RREQ, each intermediate node sets up the next hop to destination node. In that case, the **Active** field of the associated entry will be set to **True** indicating that the corresponding record can be used for data forwarding. For example, in Figure 4.2, when node $N5$ receives the RREP packet from node D, it inserts a forwarding entry to the destination D as shown in Table 4.5. Once the RREP arrives at the source S, it can start data transmission. It is noteworthy to mention that nodes which send back RREQ to previous nodes during route discovery are not considered part of the route. **This feature reduces the number of hops the data packets will travel through after a route is established between source and destination**. The dashed line in Figure 4.2 is the route created from S to D through the route setup process and it does not include nodes $N2$, $N8$ and $N9$ since these nodes sent the RREQ back to $N1$.

Route Maintenance

When an established route breaks due to a node's movement on the path, route discovery is re-initiated by the source node if it still needs a route. When either an intermediate node or the destination node moves, the source node is notified through a special control packet sent by the node at which the route broke; the source node stops sending data packets through that broken route and re-initiates a new route discovery.

4.3 Performance Analysis

In this section, we present the performance evaluation results of HGRB compared to AODV [15] and DSR [33]. We first describe the simulation environment and then discuss the simulation results. We simulated HGRB, AODV, and DSR on a variety of network topologies.

4.3.1 Simulation Environment

We used GloMoSim [88], a network-simulation tool for studying the performance of routing protocols for MANETs, for evaluating the performance of HGRB. We chose IEEE 802.11 and IP as the MAC and network layer protocols, respectively. All nodes are assumed to have fixed transmission range of 250 meters. We used the implementation of AODV that comes with the GloMoSim 2.0.3 package to compare its performance with HGRB. We ran several simulations on a set of traffic flows. Various topologies used for simulation are shown in Table 4.6; each simulation lasted for 900 seconds of simulated time. The nodes were distributed uniformly at random in the network area.

Table 4.6: Topology used for Simulation

Nodes	Network Area	CBR Flows	Packets Sent
50	1500m X 300m	30	8780
50	2000m X 2000m	30	8780
100	2500m X 2500m	30	8780
{50,100,150,200,250}	1500m X 1500m	30	8780
50	1500m X 300m	20	5168
50	2000m X 2000m	20	5168
100	2500m X 2500m	20	5168
{50,100,150,200,250}	1500m X 1500m	20	5168

4.3.2 Performance Metrics

We used the following three metrics to evaluate the performance of HGRB under various mobility model, node density, and network diameter.

1. **Packet Delivery Ratio:** The ratio of the data packets delivered to the destinations to those generated by the Constant Bit Rate (CBR) flow sources.
2. **Hop Count:** Average number of hops a data packet traverses to reach the destination.
3. **Control Overhead:** Total number of routing control packets sent during the entire simulation.

In this experiment, we varied the number of nodes simulated from 50 to 250. We used 20 and 30 CBR random traffic flows in the simulation. Each CBR flow sends packets at a speed of 2Kbps and uses 64-byte packets. Depending on the start time and end time of each sender in each flow, different number of packets are sent by different CBR flows. However, in each flow, each sender sends a packet every 0.25 second. Node mobility is set using random Way-point [33] model. Under this model, each node travels from its current location to a random destination at a random speed, the speed being uniformly distributed in a predefined range. After a node reaches its destination, it pauses for a predetermined amount of time and then moves to a new randomly chosen destination at a randomly chosen speed. In our simulation, the speed randomly chosen lies between 0 and 20 meters/second. In order to study how mobility affects the performance of HGRB, we selected pause times of 0, 20, 40, 60, 80, and 100 seconds. When the pause time is 0 seconds, every node moves continuously. As the pause time increases, the network approaches the characteristics of a fixed network.

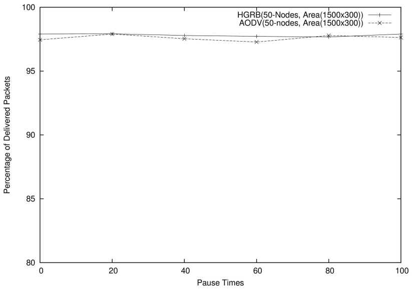


Figure 4.3: Packet Delivery Ratio As Mobility Changes (50 Nodes, 30 CBR, network area (1500m x 300m)), HGRB compared with AODV.

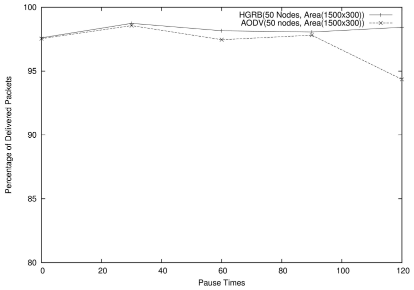


Figure 4.4: Packet Delivery Ratio As Mobility Changes (50 Nodes, 20 CBR, network area (1500m x 300m)), HGRB compared with AODV.

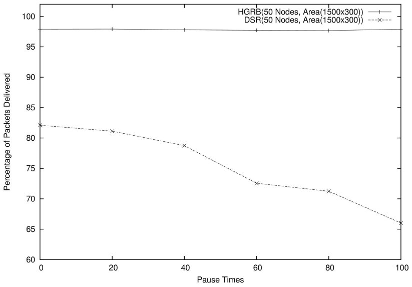


Figure 4.5: Packet Delivery Ratio As Mobility Changes (50 Nodes, 30 CBR, network area (1500m x 300m)), HGRB compared with DSR.

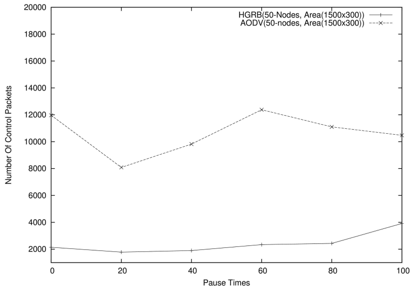


Figure 4.6: Control Overhead as Mobility Changes (50 Nodes, 30 CBR, network area (1500m x 300m)), HGRB compared with AODV.

4.3.3 Mobility

In this scenario, we evaluated HGRB with respect to the three metrics as node mobility changes from 0 to 20 meters/second. We selected CBR flows randomly; hence it is not known whether there is a valid path between the source node and the destination

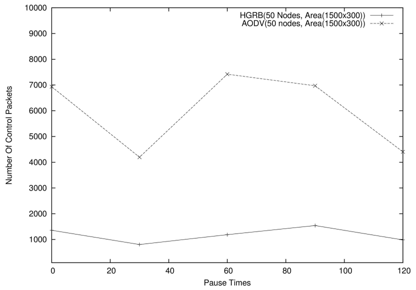


Figure 4.7: Control Overhead as Mobility Changes (50 Nodes, 20 CBR, network area (1500m x 300m)), HGRB compared with AODV.

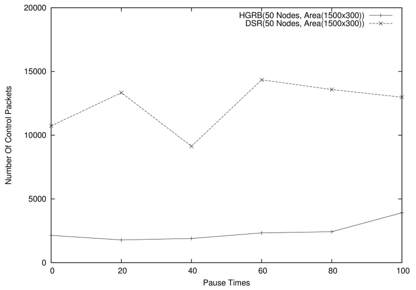


Figure 4.8: Control Overhead as Mobility Changes (50 Nodes, 30 CBR, network area (1500m x 300m)), HGRB compared with DSR.

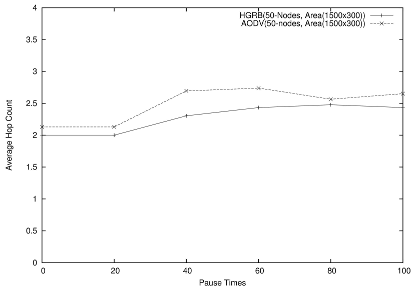


Figure 4.9: Average Hop Count as Mobility Changes (50 Nodes, 30 CBR, network area (1500m x 300m)), HGRB compared with AODV.

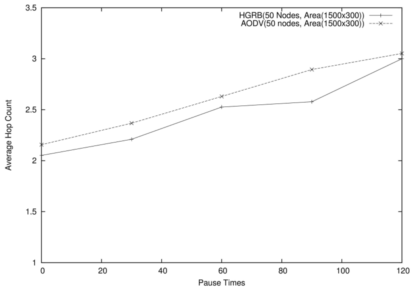


Figure 4.10: Average Hop Count as Mobility Changes (50 Nodes, 20 CBR, network area (1500m x 300m)), HGRB compared with AODV.

node for each flow. Higher number of packets (refer to Table 4.6) imposes higher demand on routing protocols as higher traffic is generated between source, destination pairs. HGRB does not broadcast RREQ packets for route establishment as in AODV and DSR. Since both AODV and DSR broadcast RREQ, the network can be overwhelmed with frequent rebroadcasting of RREQs that may collide and drop. This reduces successful establishment of routes which in turn reduces packet delivery ration. Therefore, HGRB delivers higher number of data packets than AODV for various pause times as shown in Figures 4.3 and 4.4; it delivers much higher number

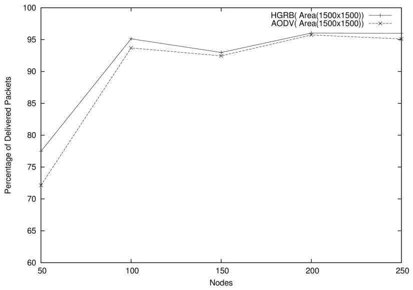


Figure 4.11: Packet Delivery Ratio as Node Density Increases (Network area (1500m x 1500m)), 30 CBR, HGRB compared with AODV.

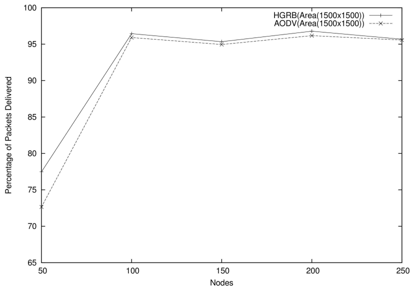


Figure 4.12: Packet Delivery Ratio as Node Density Increases (Network area (1500m x 1500m)), 20 CBR, HGRB compared with AODV.

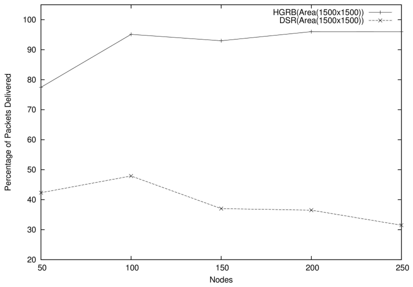


Figure 4.13: Packet Delivery Ratio as Node Density Increases (Area (1500m x 1500m)), 30 CBR, HGRB compared with DSR.

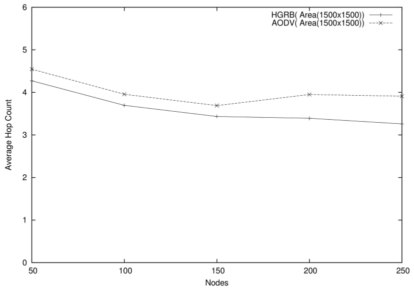


Figure 4.14: Number of Hops as Node Density Increases (Area (1500m x 1500m)), 30 CBR, HGRB compared with AODV.

of data packets than DSR for all pause times as shown in Figure 4.5.

Also, as shown in Figures 4.6, 4.7, and 4.8, control overhead for HGRB is much lower than that of AODV and DSR. This is because of the broadcast feature of AODV and DSR to establish routes. Since HGRB does not use broadcast for propagating RREQs, it incurs less control overhead compared to AODV and DSR. Figures 4.9 and 4.10 compare the number of hops on the routes through which HGRB and AODV successfully deliver data packets. Since HGRB propagates RREQs *Greedily*, the length of the routes found under HGRB are shorter especially in dense networks

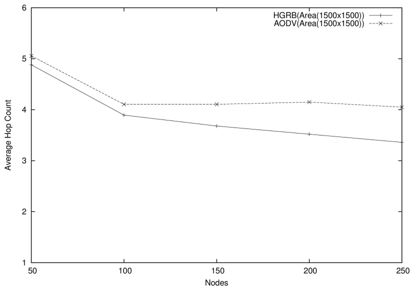


Figure 4.15: Number of Hops as Node Density Increases (Network area (1500m x 1500m)), 20 CBR, HGRB compared with AODV.

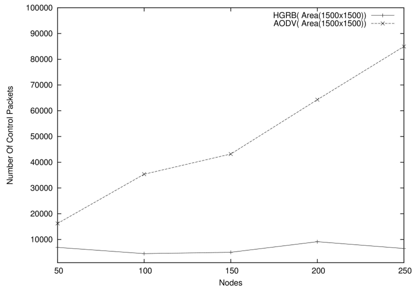


Figure 4.16: Control Overhead as Node Density Increases(Network area (1500m x 1500m)), 30 CBR, HGRB compared with AODV.

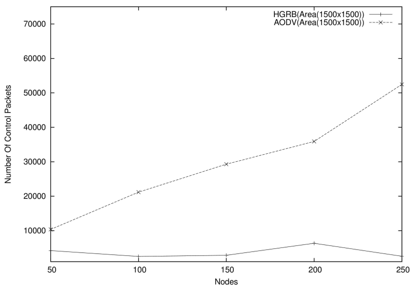


Figure 4.17: Control Overhead as Node Density Increases(Network area (1500m x 1500m)), 20 CBR, HGRB compared with AODV.

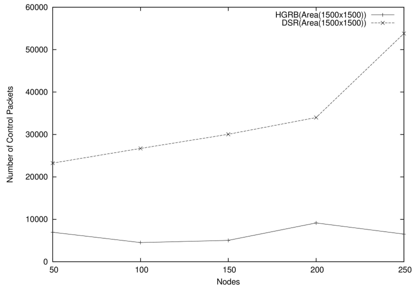


Figure 4.18: Control Overhead as Node Density Increases(Network area (1500m x 1500m)), 30 CBR, HGRB compared with DSR.

(i.e., network without voids). This makes HGRB discover routes with less number of hops on average than those found by AODV.

4.3.4 Node Density

In this scenario, to study the effect of node density on the three metrics described in 4.3.2, we varied the number of nodes from 50 to 250 in a network area of (1500m x 1500m). As node density increases, HGRB delivers higher fraction of data packets than AODV and much higher fraction of data packets than DSR as shown in Figures 4.11, 4.12, and 4.13. This is because unlike AODV and DSR, HGRB does not

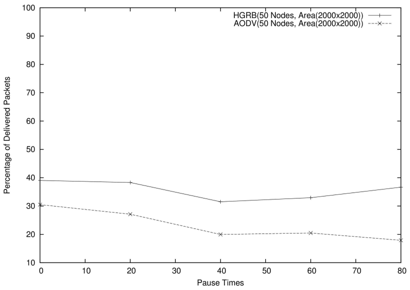


Figure 4.19: Packet Delivery Ratio in Network Area (2000m x 2000m), 30 CBR, HGRB compared with AODV.

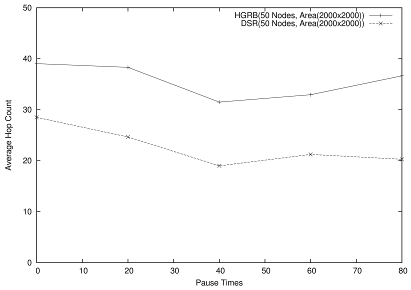


Figure 4.20: Packet Delivery Ratio in Network Area (2000m x 2000m), 30 CBR, HGRB compared with DSR.

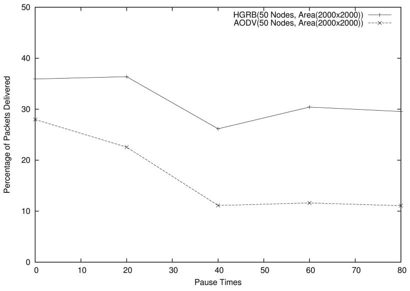


Figure 4.21: Packet Delivery Ratio in Network Area (2000m x 2000m), 20 CBR, HGRB compared with AODV.

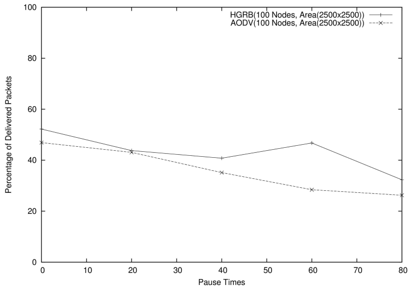


Figure 4.22: Packet Delivery Ratio in Network Area (2500m x 2500m), 30 CBR, HGRB compared with AODV.

broadcast RREQs and nodes only use their neighbor information to forward RREQs. Whereas in AODV and DSR, each node rebroadcasts the RREQ which causes high traffic especially in dense networks. Therefore, RREQs need to wait for long time before being rebroadcast by intermediate nodes. This may lead to dropping the RREQs which in turn lead to lower data packet delivery ratio than HGRB.

As shown in Figures 4.14 and 4.15, average number of hop counts on routes established by HGRB is less than those established by AODV. Since there are more voids in sparse networks, the difference in average hop count between the two algorithms is small in sparse networks. However, as the network becomes denser, number of voids

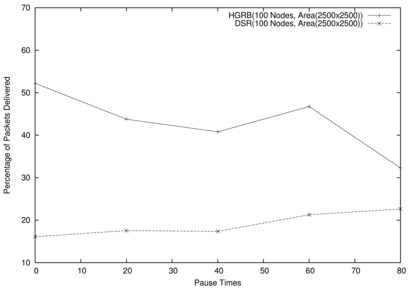


Figure 4.23: Packet Delivery Ratio in Network Area (2500m x 2500m), 30 CBR, HGRB compared with DSR.

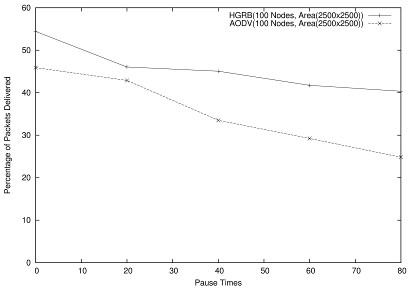


Figure 4.24: Packet Delivery Ratio in Network Area (2500m x 2500m), 20 CBR, HGRB compared with AODV.

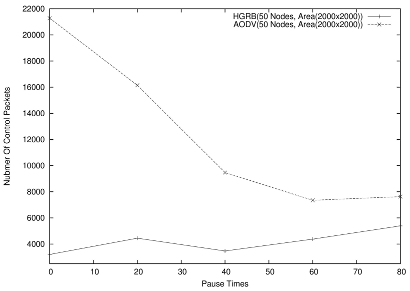


Figure 4.25: Control Overhead in Network Area (2000m x 2000m), 30 CBR, HGRB compared with AODV.

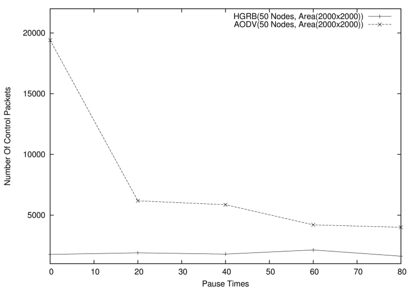


Figure 4.26: Control Overhead in Network Area (2000m x 2000m), 20 CBR, HGRB compared with AODV.

decreases and HGRB forwards RREQ packets through greedy paths; hence routes established by HGRB have significantly less number of hops than AODV in dense networks. Moreover, control overhead incurred by HGRB is less than that of AODV. As shown in Figures 4.16, 4.17, and 4.18, HGRB has almost constant control overhead as number of nodes increases from 50 to 250. On the other hand, AODV and DSR incur significantly higher control overhead than HGRB as node density increases.

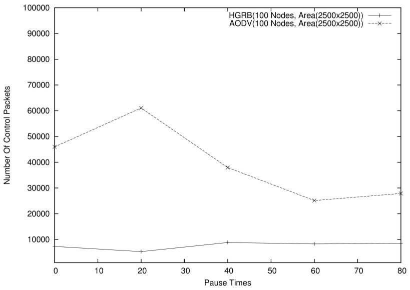


Figure 4.27: Control Overhead in Network Area (2500m x 2500m), 30 CBR, HGRB compared with AODV.

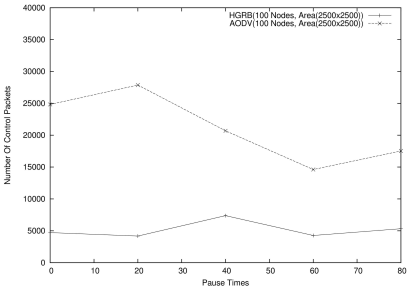


Figure 4.28: Control Overhead in Network Area (2500m x 2500m), 20 CBR, HGRB compared with AODV.

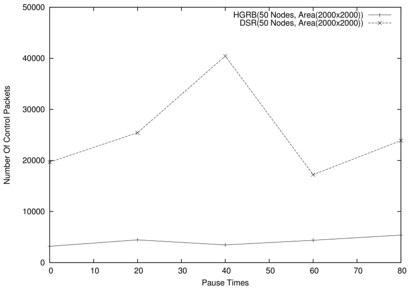


Figure 4.29: Control Overhead in Network Area (2000m x 2000m), 30 CBR, HGRB compared with DSR.

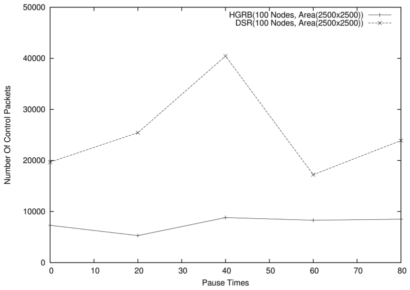


Figure 4.30: Control Overhead in Network Area (2500m x 2500m), 30 CBR, HGRB compared with DSR.

4.3.5 Network Diameter

We studied the effect of network diameter on packet delivery ratio, hop count, and control overhead. Figures 4.19, 4.20, 4.21, 4.22, 4.23, and 4.24 present packet delivery ratio results for HGRB, AODV, and DSR. Figures 4.25, 4.26, 4.27, and 4.28 show control overhead for 50-nodes and 100-nodes for HGRB and AODV. Figures 4.29 and 4.30 show the control overhead of HGRB compared with DSR for 100-nodes. Figures 4.31, 4.32, 4.33, and 4.34 present average hop count results for HGRB and AODV.

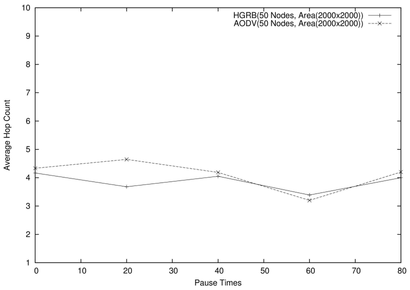


Figure 4.31: Average Hop Count in Network Area (2000m x 2000m), 30 CBR, HGRB compared with AODV.

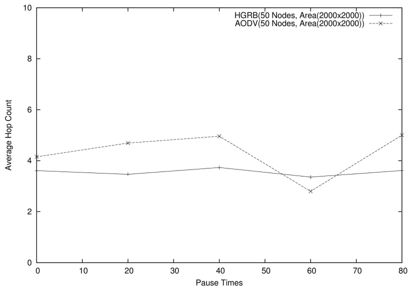


Figure 4.32: Average Hop Count in Network Area (2000m x 2000m), 20 CBR, HGRB compared with AODV.

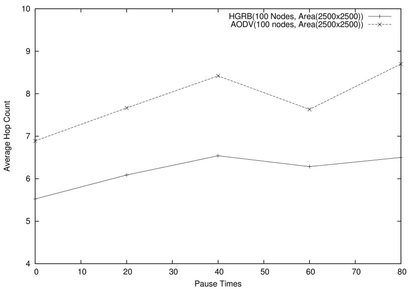


Figure 4.33: Average Hop Count in Network Area (2500m x 2500m), 30 CBR, HGRB compared with AODV.

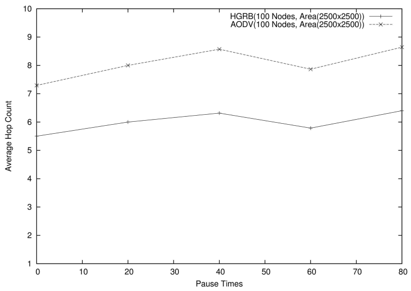


Figure 4.34: Average Hop Count in Network Area (2500m x 2500m), 20 CBR, HGRB compared with AODV.

Chapter 5 SAriadne: A Secure Source Routing Protocol that Prevents Hidden-Channel Attacks

5.1 Introduction

Nodes in a Mobile Ad Hoc Network (MANET) form a network among themselves without the use of any fixed infrastructure (i.e., no centralized administration) such as access points or base stations, and communicate with each other by cooperatively forwarding packets on behalf of others. MANETs have applications in areas such as military, disaster rescue operations, monitoring animal habitats, etc. where establishing communication infrastructure is not feasible [16, 23, 27, 24, 25].

Routing protocols designed for MANETs can be broadly classified as geographic routing protocols (e.g., GPSR [8]) and topology-based routing protocols. Topology-based routing can be classified into proactive routing protocols like DSDV [13] in which nodes use pre-established table-based routes [14], reactive (on-demand) routing protocols (e.g., AODV [15]), and hybrid routing protocols [16, 89, 19].

There are two main functions in routing: route discovery and data forwarding. The first one is concerned with finding routes between nodes, while the latter uses these routes to forward data packets. Informally, a secure routing protocol enables nodes to exchange control information and data in the presence of adversaries whose objective is to disrupt the functioning of the routing protocol. Several mechanisms have been proposed to provide secure routing in MANETs ([20] presents a survey of secure routing protocols).

Due to their special characteristics, MANETs are more vulnerable to attacks than infrastructure based networks. There are two main categories of attacks, namely, passive attacks and active attacks [90]. In passive attacks, adversarial nodes do not disrupt the operation of routing protocols, whereas in active attacks, adversarial nodes disrupt network operations as well as they perform effective violations. They can control the flow of the network traffic by injecting incorrect control information during route discovery. Adversarial nodes can disrupt at different stages including route discovery phase, route maintenance phase, and data forwarding phase.

An adversarial node can disrupt the route discovery by offering a better route than a route offered by a genuine node to disrupt packet delivery. Under this type of attack, a malicious node modifies routing control information or specific routing metrics. Another type of attack is done using impersonation (i.e., spoofing) wherein adversarial nodes hide their identities (e.g., real IP address or MAC address) and use other identities to launch sophisticated attacks.

5.1.1 Objectives

Our focus is mainly on security issues related to source routing protocols (e.g., SRP [46], Ariadne [21], endairA [3], etc.). As we point out in Section 5.2, both Ariadne [21] and endairA [3] are prone to hidden channel attacks that target the route discovery. In this chapter, we propose a novel secure source routing protocol called *Secure Ariadne (SAriadne)* that establishes a secure and valid route. SAriadne uses *sanitizable signatures* [91] to prevent these attacks.

The rest of the chapter is organized as follows: In Section 5.2, we present the hidden channel attacks on the related protocols. In Section 5.3 we discuss *Chameleon Hash* functions and *Sanitizable Signatures* which are needed in the design of the proposed protocol. In Section 5.4, we present our protocol. In Section 5.5, we analyze and discuss how our protocol prevents hidden channel attacks.

5.2 Hidden Channel Attacks on the Discussed Source Routing Protocols

In this section, we present a brief review of the attacks against the algorithms presented in Section 2.5.

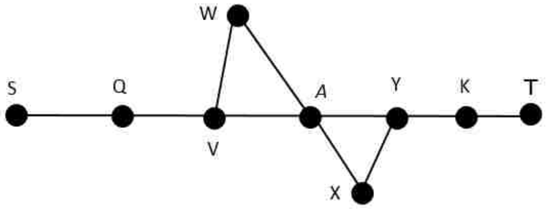


Figure 5.1: A Sample Network Configuration Wherein an Attack Exists Under SRP [4].

5.2.1 Attack on SRP

In this section, we discuss how SRP is prone to hidden channel attack discovered in [4]. We use the network configuration in Figure 5.1 to illustrate how SRP is prone to hidden channel attack, in which A is an adversarial node. Let us assume that the node S in Figure 5.1 initiates a route request to discover a route to the target T. When V receives the request for the first time, it rebroadcasts the request. Thus, A receives the request m_V from V, where

$$m_V = (rreq, S, T, id, sn, (Q, V), mac_S),$$

where id is the request id, sn is a sequence number, and mac_S is the MAC computed by S using the key it shares with T. A, the adversaial node, then inserts an arbitrary sequence of identifiers λ into the request and broadcasts it in the name of X as follows

$$m_A = (rreq, S, T, id, sn, (Q, V, W, \lambda, X), mac_S).$$

The adversarial node A cannot guarantee that the pair V and W and the pair X and Y are neighbors; however, since the nodes (i.e., V,W,X, and Y) in these pairs are neighbors of A, A makes a guess that V and W, as well as X and Y are neighbors. Y is a neighbor of both A and X and the node list ends with X; hence, when Y receives m_A, Y appends its id to the node list in the request, and rebroadcasts the updated request m_Y, where

$$m_Y = (rreq, S, T, id, sn, (Q, V, W, \lambda, X, Y), mac_S).$$

When the target T receives the request from K, after it successfully verifies mac_S, T computes its MAC (i.e., mac_T) over the route and unicasts the reply m_T via the nodes in the route request in the reverse order where

$$m_T = (rrep, S, T, id, sn, (Q, V, W, \lambda, X, Y, K), mac_T).$$

When the intermediate node Y forwards m_T to node X, node A overhears the transmission. A then forwards the message m_T to V in the name of W. Since W and V are neighbors, V believes the message is from W and forwards it to Q which in turn sends it to S. S then successfully verifies mac_T and accepts the route $(Q, V, W, \lambda, X, Y, K)$ which is clearly an invalid route.

Thus, the SRP is susceptible to hidden channel attack.

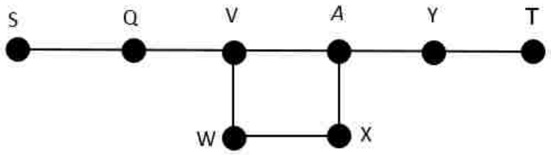


Figure 5.2: A Sample Network Configuration Wherein an Attack Exists on Ariadne with Signatures [4].

5.2.2 Attack on Ariadne with Signatures

Buttyán and Vajda [4] discovered an attack against Ariadne with signatures presented in Section 2.5.3. Figure 5.2 shows part of the network configuration wherein the adversarial node A launches an attack on the route discovery. Suppose that the source S (in Figure 5.2) sends a route request towards the target T. A receives the request m_V from node V, where

$$m_V = (rreq, S, T, id, h_V, (Q, V), (sig_Q, sig_V)),$$

and does not rebroadcast the request. Also, A receives request m_X from node X, where

$$m_X = (rreq, S, T, id, h_X, (Q, V, W, X), (sig_Q, sig_V, sig_W, sig_X)).$$

A obtains h_V from m_V and the signatures sig_Q, sig_V, and sig_W from m_X. A also knows that W and V are neighbors from m_X. A then computes its per-hop hash value (i.e., $h_A = H(A, H(W, h_V))$), where H is the publicly known hash function. A then uses this information to generate and broadcast the request m_A, where

$$m_A = (rreq, S, T, id, h_A, (Q, V, W, A), (sig_Q, sig_V, sig_W, sig_A)).$$

When the target T receives the request, T verifies the signatures and creates a route reply m_T and sends it back to S, where

$$m_T = (rrep, T, S, (Q, V, W, A, Y), (sig_Q, sig_V, sig_W, sig_A, sig_Y, sig_T)).$$

When A receives m_T, it forwards it to V in the name of W which in turn forwards it to Q. When S receives the reply, it validates all the signatures and accepts the route (Q, V, W, A, Y), which is an invalid route because A and W are not neighbors.

Thus, Ariadne with signature protocol is susceptible to hidden channel attack.

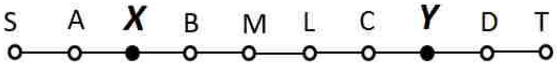


Figure 5.3: A Sample Network Configuration wherein an Attack Exists on Ariadne with MAC[3].

5.2.3 Attack on Ariadne with MAC

Ács et al. [3] found an attack on Ariadne with MAC presented in Section 2.5.4. Consider the network configuration shown in Figure 5.3, wherein S initiates a route request to find a route to the target T, and X and Y are adversarial nodes that collude. X, the first adversarial node, receives the request m_A, where

$$m_A = (rreq, S, T, id, h_A, (A), (mac_A)).$$

Instead of appending its MAC, X puts h_A on the MAC list and rebroadcasts the request m_X. X does that because Y will need this hash value to omit the nodes between X and Y.

$$m_X = (rreq, S, T, id, h_A, (A, X), (mac_A, h_A)).$$

Because intermediate nodes do not verify the MACs, the intermediate node B does not detect this attack. The second adversarial node Y receives the route request m_C, where

$$m_C = (rreq, S, T, id, H(C, H(L, H(M, H(B, h_A)))), (A, X, B, M, L, C),$$

$$(mac_A, h_A, mac_B, mac_M, mac_L, mac_C)).$$

Then, Y drops the nodes (B, M, L, C) from the node list and their corresponding MACs from the MAC list (i.e., $(mac_B, mac_M, mac_L, mac_C))$. Y can do that because it knows that the request has passed through the first adversarial node X by recognizing X in the node list. Y also could get h_A from the MAC list by determining the position of X in the node list. Hence, Y computes $h_Y = H(Y, h_A)$, needed to omit the nodes (B, M, L, C), and its MAC mac_Y on the modified request. Then Y rebroadcasts the modified request m_Y that is received, updated, and rebroadcast by D, where

$$m_Y = (rreq, S, T, id, h_Y, (A, Y), (mac_A, mac_Y)).$$

When T receives the request, it validates the MACs and the per-hop hash value in the request, generates a reply m_T, and sends it back to S, where

$$m_T = (rrep, S, T, (A, Y, D, T), mac_T).$$

When Y receives m_T, it re-inserts the dropped nodes into the node list and forwards the modified reply m_{Yrrep} to C

$$m_{Yrrep} = (rrep, S, T, (A, X, B, M, L, C, Y, D, T), mac_T).$$

Since the intermediate nodes B, M, L, C do not verify m_{Yrrep}, each of them forwards the reply m_{Yrrep}. When X receives the reply, it removes the nodes B, M, L, C and its id from the list and forwards the reply $m_{X_{rrep}}$ to A, where

$$m_{X_{rrep}} = (rrep, S, T, (A, Y, D, T), mac_T).$$

When S receives the $m_{X_{rrep}}$, it computes the MAC over the fields preceding the mac_T in route reply and verifies if it is same as mac_T that was computed by T; if it is, then it accepts the route as a valid route, even though it is an invalid route.

Thus, the Ariadne with MAC protocol is susceptible to hidden channel attack.

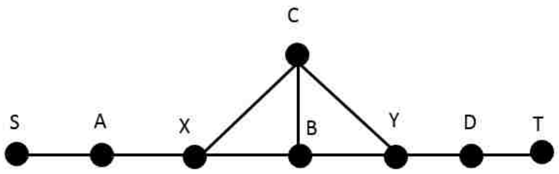


Figure 5.4: A Sample Network Configuration wherein an Attack on the Optimized Version of Ariadne with Iterated MAC Exists.

5.2.4 An Attack on the Optimized Version of Ariadne with Iterated MAC

In this section, we discuss how optimized version of Ariadne with iterated MAC is prone to attack from adversarial nodes. This attack is first presented in [3], and briefly described in [92]. Consider the network configuration in Figure 5.4, in which the source S needs to find a route to the target T, and X and Y are adversarial nodes that collude to launch and attack. The first adversarial node X receives a route request m_A from the node A, where

$$m_A = (rreq, S, T, id, (A), mac_{SA}),$$

X computes its MAC (i.e., mac_{SAX}) over m_A after appending its id to the node list in the request and then broadcasts the updated request m_X, where

$$m_X = (rreq, S, T, id, (A, X), mac_{SAX}).$$

When B and C receive m_X, they update the request and broadcast the corresponding route request. Y does not respond to either requests coming from B or C. A little later, X creates a route reply $m_{X_{rrep}}$ in the name of Y and unicasts it to B, where

$$m_{X_{rrep}} = (rrep, S, T, id, (A, X, B, Y), mac_{SAX}).$$

Note that the MAC in this fake reply is wrong (i.e., it was not computed by the target). B only checks the id of the reply and that both X and Y are its neighbors. Since B has previously received request m_X with id same as that included in this reply $m_{X_{rrep}}$, it retransmits $m_{X_{rrep}}$ to X. Y intercepts $m_{X_{rrep}}$ whose MAC is mac_{SAX} which is needed by Y to remove B. Y then creates and broadcasts route request m_Y, where

$$m_Y = (rreq, S, T, id, (A, X, Y), mac_{SAXY}).$$

Node D receives this request, appends its label to the node list, computes its MAC, and rebroadcasts the updated request m_D, where

$$m_D = (rreq, S, T, id, (A, X, Y, D), mac_{SAXYD}).$$

T receives m_D and verifies the iterated MAC mac_{SAXYD}. Since mac_{SAXYD} was correctly constructed, T accepts it, generates a route reply m_T, and sends it back to the source S, where

$$m_T = (rrep, S, T, id, (A, X, Y, D), mac_T).$$

When D receives m_T, it accepts it because D's label is in the list and both T and Y are neighbors of D. When Y receives m_T from D, it adds the label for C to the node list and sends the modified message $m_{Y_{rrep}}$ to C, where

$$m_{Y_{rrep}} = (rrep, S, T, id, (A, X, C, Y, D), mac_T).$$

Node C accepts the message and forwards it to X. When X receives $m_{Y_{rrep}}$, it removes the label of C from the node list and forwards the modified message to S which in turn successfully verifies the reply. However, the verified reply does not contain a valid route.

In this attack, to remove the node B or C from the node list, the second adversarial node Y needed the mac_{SAX} which was computed by X in the message $m_{X_{rrep}}$. Y needs this MAC in order to compute mac_{SAXY}. Therefore, the adversarial nodes X and Y successfully shortened the route to become (A, X, Y, D) which is invalid because X and Y are not neighbors.

Thus, the optimized version of Ariadne with iterated MAC is susceptible to hidden channel attack.

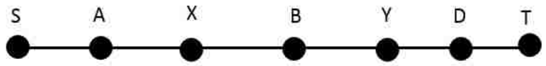


Figure 5.5: A Sample Network Configuration Wherein an Attack Exists in endairA.

5.2.5 Attack on endairA

Mike et al. [92] found an attack on endairA [3], presented in Section 2.5.6. Consider the network configuration shown in Figure 5.5 in which the source S needs to find a route to the target T. In this configuration, nodes X and Y are adversarial nodes and they collude to remove B from the node list of the route request. Under endairA, the second adversarial node Y receives a route request m_B from node B, where

$$m_B = (rreq, S, T, id, (A, X, B)).$$

Since endairA does not authenticate route requests, Y drops node B from the node list contained in m_B and boadcasts the modified route request m_Y,where

$$m_Y = (rreq, S, T, id, (A, X, Y)).$$

When T receives the route request, it generates a secured route reply and unicasts it back to S. The second adversarial node Y receives the reply $m_{D_{rrep}}$ from D, where

$$m_{D_{rrep}} = (rrep, S, T, id, (A, X, Y, D), (sig_T, sig_D)).$$

If Y signs the reply and forwards it to B, B will discard it because its id is not in the node list. Also, S validates the route reply only if every node in the node list returned in route reply has signed the reply. Therefore, the first adversarial node

X needs the signature list (sig_T, sig_D, sig_Y). Since X and Y are adversarial nodes, they could have shared their private information. This means that X can reconstruct the digital signature of Y (i.e., sig_Y) . Now, Y needs to find a mechanism through which it can deliver the signatures sig_T and sig_D to X. Suppose that the node D had previously issued a route request $m_{D_{rreq}}$ with a request id $id^{'}$ to discover a route to node A. Therefore, Y must have received $m_{D_{rreq}}$ from D

$$m_{D_{rreq}} = (rreq, D, A, id^{'}, ()).$$

Since route requests are not secured, Y can exploit the previous request from D to A to deliver the signatures (i.e., (sig_T and sig_D)) to X. To do that, Y modifies the id $id^{'}$ of the request $m_{D_{rreq}}$ into some other id $id^{''}$ such that it contains information that X can use to reconstruct the signatures sig_T and sig_D. *This information can also be encrypted.* Intermediate nodes do not detect this modification and rebroadcast the request. When X receives this request, it reconstructs the signatures, generates route reply m_X, and forwards it to A which in turn signs the reply and forwards it to S, where

$$m_X = (rrep, S, T, id, (A, X, Y, D), (sig_T, sig_D, sig_Y, sig_X)).$$

The source S then verifies that each individual signature was calculated correctly by the corresponding nodes and accepts (A,X,Y,D) as a valid route.

Thus, endairA protocol is susceptible to hidden channel attack.

5.3 Preliminaries

In this section, we review Chameleon Hash functions and Sanitizable Signatures needed in the design of the proposed protocol.

5.3.1 Chameleon Hash Functions without Key Exposure

Chameleon hash functions have the same characteristics of any cryptographic hash function with a trapdoor property, which allows the computation of collisions and second pre-images once the trapdoor information is known.

Ateniese and Medeiros [93] proposed a chameleon hash with a key exposure freeness property by introducing a label ($\mathcal{L}$) which is fully committed to the signature together with the original identity of the recipient to receive the message from the signer. The key exposure free chameleon hash function is defined by the following set of algorithms:

- *Key Generation (KeyGen)*: A probabilistic algorithm that accepts a security parameter κ as input, and outputs a (secret key, public key) pair (*sk*,*pk*) as follows

$$\mathbf{KeyGen} : 1^{\kappa} \longrightarrow (sk, pk))$$

- *Hash*: A function that accepts a public key *pk*, a label $\mathcal{L}$, a message m and an auxiliary random parameter r as input, and outputs a bitstring C of fixed length τ as follows

$$\mathbf{Hash} : (pk, \mathcal{L}, m, r) \longrightarrow C \in \{0,1\}^{\tau}$$

- *Universal Forge (Uforge)*: A function that accepts the secret key *sk* (associated with public key *pk*), a label $\mathcal{L}$, a message m and an auxiliary parameter r as input, and computes another message m' and random parameter r' as follows

$$\mathbf{UForge}(sk, \mathcal{L}, m, r) \longrightarrow (m', r')$$

such that

$$\textbf{Hash}(pk, \mathcal{L}, m, r) = C = \textbf{Hash}(pk, \mathcal{L}, m', r')$$

- *Instance Forge (IForge)*: A function that accepts a public key pk, a label $\mathcal{L}$, and two pairs of a message and an auxiliary random parameter (m, r, m', r') as input, and computes another collision message m'' and random parameter r'' as follows

  $$\textbf{IForge}(pk, \mathcal{L}, m, r, m', r') \longrightarrow (m'', r'')$$

  such that

  $$\textbf{Hash}(pk, \mathcal{L}, m, r) = C = \textbf{Hash}(pk, \mathcal{L}, m', r') = \textbf{Hash}(pk, \mathcal{L}, m'', r'')$$

The above mentioned function ensures the following security requirements:

- *Collision-Resistance,* which means that given only pk, $\mathcal{L}$, m and r, there is no efficient algorithm that can find a second pair m', r' such that $C = Hash(pk, \mathcal{L}, m, r) = Hash(pk, \mathcal{L}, m', r')$.

- *Semantic Security,* which means that the chameleon hash value C does not reveal anything about the possible message m that was hashed.

- *Key Exposure Freeness,* which means that given $C = Hash(pk, \mathcal{L}, m, r)$ there is no efficient algorithm that can find a collision (i.e., a second pair m', r' mapping to the same digest C) if a recipient with public key pk has never computed a collision under label $\mathcal{L}$.

Now, we describe the algorithms which create practical instances by using cryptographic primitives from a discrete log-based (DL) cryptosystem to construct an efficient chameleon hash trapdoor commitment scheme providing key exposure freeness as presented in [93].

The scheme consists of three efficient algorithms: Key Generation Algorithm, Hash Calculation Algorithm, and Collision Finding Algorithm.

Key Generation Algorithm

An algorithm that takes a security parameter λ as input and outputs system public parameters $\mathsf{params} = \langle p, q, g, H \rangle$, where

- p and q are primes, $p = 2q + 1$,
- g is a generator of the subgroup of quadratic residues Q_p of $\mathbb{Z}_p^*$,
- and $H : \{0,1\}^* \mapsto \{0,1\}^\tau$ is a collision-resistant hash function mapping arbitrary-length bitstrings to strings of fixed length τ.

The recipient chooses as secret key x at random in [1,q-1], and computes the corresponding public key as $y = g^x$.

Hash Calculation Algorithm

An algorithm that takes a public key y, message m, and random values (r, s) $\in$ $\mathbb{Z}_q$ x $\mathbb{Z}_q$ then computes $e = H(m, r)$ and $C \leftarrow Hash(m, r, s) = r - (y^e g^s \, mod \, p) \, mod \, q$.

Collision Finding Algorithm

An algorithm that takes a hash output C, a random message m', and a random value $k' \in [1, q-1]$ then computes a collision (m', r', s') as follows

$r' = C + (g^{k'} \ mod \ p) \ mod \ q$

$e' = H(m', r')$

$s' = k' - e'^{x} \ mod \ q$

such that $C = Hash(m, r, s) = Hash(m', r', s')$

5.3.2 Sanitizable Signatures

The Sanitizable Signature scheme proposed by Ateniese el al. [91] allows a node to modify designated portions of the document and produce a valid signature on the legitimately modified document without any help from the original signer. The designated portions are indicated as mutable which are subject to a prior agreement between the signer and the node. However, a sanitizable signature scheme can be weakly transparent which means verifiers can identify which parts of the message are potentially sanitizable and, consequently, which parts are immutable.

The terms 'censor' and 'target' are used interchangeably in the rest of the chapter.

A sanitizable signature scheme is defined by the following set of algorithms:

- *Key Generation Algorithm*: A probabilistic algorithm to compute the Signer's two public-private key pairs: pk^{S}_{sign} and sk^{S}_{sign} (for a standard digital signature algorithm) and the Censor's two public-private key pairs pk^{T}_{sanit} and sk^{T}_{sanit} (for sanitization steps).

- *Signing Algorithm*: A deterministic algorithm which takes as input a message m, a private signing key sk^{S}_{sign}, a public sanitization key pk^{T}_{sanit}, random coin r, and produces a signature σ as follows

$$\sigma \leftarrow SIGN(m, r; sk^{S}_{sign}, pk^{T}_{sanit})$$

- *Verification Algorithm*: A deterministic algorithm that, on input of a message m, a possibly valid signature σ on m, a public signing key pk_{sign}^{S} and a sanitization key pk_{sanit}^{T}, outputs TRUE or FALSE as follows

$$VERIFY(m, \sigma; pk_{sign}^{S}, pk_{sanit}^{T}) \rightarrow \{TRUE, FALSE\}$$

- *Sanitization Algorithm*: A deterministic algorithm that, on input of a message m, a signature σ on m using public signing key pk_{sign}^{S}, a private sanitizing key sk_{sanit}^{T}, and a new message m', produces a new signature σ' on m' as follows

$$\sigma' \leftarrow SANIT(m, \sigma, m'; pk_{sign}^{S}, sk_{sanit}^{T})$$

The above mentioned scheme meets the following security requirements:

- *Correctness* which means that the verification algorithm must accept a signature produced by the signing algorithm.

- *Unforgeability* which means that it is difficult to produce a valid signature on a message that verifies against the associated public key without the knowledge of the private signing key.

- *Identical Distribution* which means that σ values produced by the sanitization algorithm are distributed identically to those produced by the signing algorithm.

5.4 The Proposed Protocol

In this section, we present our secure on-demand routing protocol based on Sanitizable Signatures called *SAriande*. This protocol prevents the *Hidden Channel Attacks* on the optimized version of Ariadne and the endairA protocol presented in Sections 5.2.4

and 5.2.5, respectively. In the following subsections, we present the details of our protocol.

5.4.1 Protocol Setup

In our protocol, we use the sanitizable signature [91] constructed based on the chameleon hashes presented in [93]. The two pairs of keys used in our protocol are the (private, public) signing keys $(sk^S_{sign}, pk^S_{sign})$ of the source S, and the (private, public) sanitizing keys $(sk^T_{sanit}, pk^T_{sanit})$ of the target T. To avoid confusion, $SIGN(.)$ is used for the sanitizable signature and $Sig(.)$ is used for the underlying signature algorithm. We assume that every node has the public key of both the signer and sanitizer. The basic idea behind the route request propagation and the route reply propagation of our protocol is similar to that in the optimized version of Ariadne, but we prevent hidden channel attacks using sanitizable signature.

5.4.2 Basic Idea Behind the Protocol

There are different attacks and security issues associated with routing in MANETs (e.g., black hole attacks, Gray hole attacks, jamming attacks, blackmail attacks, Byzantine attacks, overwhelming attacks, hidden channel attacks, etc.) [94, 21]. However, our main focus in this chapter is to design a routing protocol that prevents *hidden channel attacks* discussed in Section 2.5. We mainly consider the cases where adversarial nodes are not neighbors (e.g., the attack shown in Section 5.2.4).

We use *sanitizable signature* [91] to detect and prevent hidden channel attacks. The source node signs the route request using its private key sk^S_{sign} and the public sanitizing key pk^T_{sanit} of the target. This sanitizable signature is weakly transparent [91] to all the nodes in the network. It will be sanitized by the target T using T's private sanitizing key sk^T_{sanit}.

Since this signature is weakly transparent, every other node can verify modifications in the sanitizable portion of the message. This helps every node verify whether the reply is coming from the target node T. Therefore, no malicious node can impersonate the target T. This prevents the hidden channel attack during the propagation of the route request to the target node.

5.4.3 Basic Route Discovery

When a source node S needs to discover a route to a target T, it broadcasts a route request. To do that, S generates a message m and a signature σ on m using an algorithm called $SIGN$, described in Section 5.3.2. The route request message M_S initiated by S and targeted to T contains eight fields: *(route request, message m, sanitizable signature σ, source, target, request id, node list, MAC list).* The *source* and the *target* fields are set to the addresses or identifiers of the source and the target nodes, respectively. The *request id* identifies the route request. As in Ariadne [21], the source S initializes the *node list* and *MAC list* to empty lists. S creates a message $m = (m_1, ..., m_t)$ that is partitioned into t blocks for some constant t and initializes m to *(request id, message id, source, target, sanitizable-portion of the message)*, where

$$m = (id, m_{ID}, S, T, sanitizable - portion).$$

The source S decides which portions, say $m_{i_1}, ..., m_{i_k}$, can be modified by the target T. The source computes the chameleon hash, $CH_{pk^T_{sanit}}(.)$, using the target's public key pk^T_{sanit} for all m_i where $i \in \{1, ..., t\}$ as follows

$$\bar{m}_i = \begin{cases} CH_{pk^T_{sanit}}(m_{ID}||i||m_i, r_i)\text{if } i \in \{i_1, i_2, ..., i_k\}, \\ \\ m_i||i \text{ otherwise} \end{cases}$$

To sign $\bar{m}$, where $\bar{m} = (\bar{m}_1, \bar{m}_2, ..., \bar{m}_t)$, S uses a *Signing Algorithm* similar to that presented in Section 5.3.2. S signs $\bar{m}$ with its private signing key sk^S_{sign} and the public sanitization key pk^T_{sanit} of the target T. The $SIGN$ algorithm takes $\bar{m}$, the private signing key of the source sk^S_{sign}, the public sanitizaiton key of the target pk^T_{sanit}, and a random coin r as input and produces the signature σ, where

$$\sigma = SIGN(\bar{m}, r; sk^S_{sign}, pk^T_{sanit}) = Sig_{sk^S_{sign}}(m_{ID}||t||pk^T_{sanit}||\bar{m}_1||...||\bar{m}_t).$$

Then S broadcasts the route request M_S, where

$$M_S = (rreq, \bar{m}, \sigma, S, T, id, (), ()).$$

The route request M_j forwarded by an intermediate node X_j, $1 \leq j \leq n$, on the route $S = X_0$,X_1,...,X_n,$X_{n+1} = T$ is of the form

$$M_j = (rreq, \bar{m}, \sigma, S, T, id, (X_1, ..., X_j), (mac_{SX_1 \cdots X_j})).$$

$mac_{SX_1 \cdots X_j}$ is the MAC computed by X_j with the key it shares with T on the route request M_{j-1} received from X_{j-1}, where

$$M_{j-1} = (rreq, \bar{m}, \sigma, S, T, id, (X_1, ..., X_{j-1}), (mac_{SX_1 \cdots X_{j-1}})).$$

When the intermediate node X_j receives the route request that is intended to a target T, X_j verifies the signature σ using a Verification Algorithm, $VERIFY$, explained in Section 5.3.2, where

$$VERIFY(\bar{m}, \sigma; pk^S_{sign}, pk^T_{sanit}) \longrightarrow \{TRUE, FALSE\}.$$

This verification is necessary to prevent the hidden channel attack on endairA mentioned in Section 5.2.5. If the signature is valid, X_j checks whether it has received the same request by checking the *source id*, *target id*, and *request id.* It discards the request if it has been received before. If not, X_j appends its address X_j to the $Node\ List$ and replaces the $mac_{SX_1 \cdots X_{j-1}}$ with $mac_{SX_1 \cdots X_j}$ as explained above; X_j then rebroadcasts the route request M_j.

When the target T receives the route request, first it verifies the signature σ using the verification algorithm *VERIFY*, and then it validates the nodes in the list by recomputing all the intermediate MAC values. If these verifications are successful, T sanitizes the message $\bar{m}$ using its private sanitization key sk^T_{sanit} (as explained in Section 5.3.2) and generates a message $\bar{m}'$. For all $\bar{m}_i$ where i in $\{1, ..., t\}$,

$$\bar{m}' = \begin{cases} CH_{sk^T_{sanit}}(m_{ID}||i||m'_i, r'_i) \text{ for } i \in \{i_1, i_2, ..., i_k\}, \\ \\ \bar{m}_i||i \text{ otherwise} \end{cases}$$

The mutable portion of the message m will be replaced with the nodes in the $Node\ List.$

The sanitizing algorithm $SANIT$ takes as input $\bar{m}$, the signature σ on $\bar{m}$, the modified message $\bar{m}'$, the public signing key pk^S_{sign} of the source S, and the private sanitizing key sk^T_{sanit} of the target T to generate the sanitized signature σ', where

$$\sigma' \leftarrow SANIT(\bar{m}, \sigma, \bar{m}'; pk^S_{sign}, sk^T_{sanit}).$$

T then generates a route reply message M_T which consists of the *ids* of source and target, the *request id*, the *Node List* (i.e., the accumulated route obtained from the request), the modified message $\bar{m}'$, and the sanitized signature σ', where

$$M_T = (rrep, \bar{m}', \sigma', S, T, id, (X_1, ..., X_n)).$$

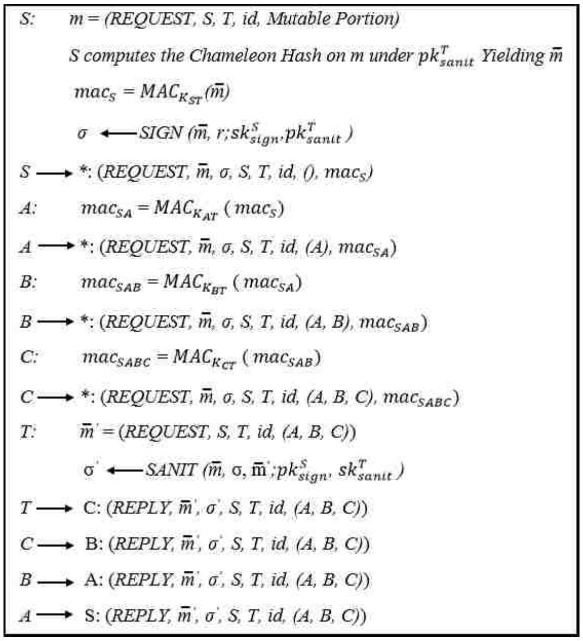
Unlike Ariadne [21], the target does not compute the MAC over the reply because the reply is authenticated using sanitized signature σ'. The reply is then sent back to S on the reverse route included in the route request. When each node X_j in the *Node List* receives the reply, it verifies the sanitized signature σ' of the target, using

$$VERIFY(\bar{m}', \sigma'; pk^S_{sign}, pk^T_{sanit}) \longrightarrow \{TRUE, FALSE\}.$$

If this test returns $FALSE$, X_j discards the reply. If it returns $TRUE$, X_j verifies that the mutable portion of the message has been modified (i.e., the signature has been sanitized) by the target. If not, then X_j discards the reply. If both these tests succeed, X_j unicasts the reply message to X_{j-1}, $1 \leq j \leq n$, on the reverse route $T = X_{n+1}, X_n$,X_{n-1},...,X_1,$X_0 = S$. Weak transparency of the sanitizable signature gives intermediate nodes the ability to verify whether the signature has been sanitized by T.

When S receives the reply, it checks whether the sanitized signature σ' passes the verification using *VERIFY*; if so, the source accepts the route; otherwise, it discards it. Unlike Ariadne [21], the source does not need to recalculate the corresponding MAC of each intermediate node since the authentication is done through the sanitized signature σ'.

Figure 5.6 shows the steps involved in route discovery under our protocol when the source tries to discover a route to the target node T and the route request goes through the intermediate nodes *(A,B,C)* before reaching T. Figures 5.7 and 5.8 respectively show the formal descriptions of how both route requests and route replies at intermediate nodes are processed in our protocol.



S: $m = (REQUEST, S, T, id, Mutable\ Portion)$

S computes the Chameleon Hash on m under pk^{T}_{sanit} Yielding $\bar{m}$

$mac_S = MAC_{K_{ST}}(\bar{m})$

$\sigma \longleftarrow SIGN(\bar{m}, r; sk^{S}_{sign}, pk^{T}_{sanit})$

$S \longrightarrow *: (REQUEST, \bar{m}, \sigma, S, T, id, (), mac_S)$

A: $mac_{SA} = MAC_{K_{AT}}(mac_S)$

$A \longrightarrow *: (REQUEST, \bar{m}, \sigma, S, T, id, (A), mac_{SA})$

B: $mac_{SAB} = MAC_{K_{BT}}(mac_{SA})$

$B \longrightarrow *: (REQUEST, \bar{m}, \sigma, S, T, id, (A, B), mac_{SAB})$

C: $mac_{SABC} = MAC_{K_{CT}}(mac_{SAB})$

$C \longrightarrow *: (REQUEST, \bar{m}, \sigma, S, T, id, (A, B, C), mac_{SABC})$

T: $\bar{m}' = (REQUEST, S, T, id, (A, B, C))$

$\sigma' \longleftarrow SANIT(\bar{m}, \sigma, \bar{m}'; pk^{S}_{sign}, sk^{T}_{sanit})$

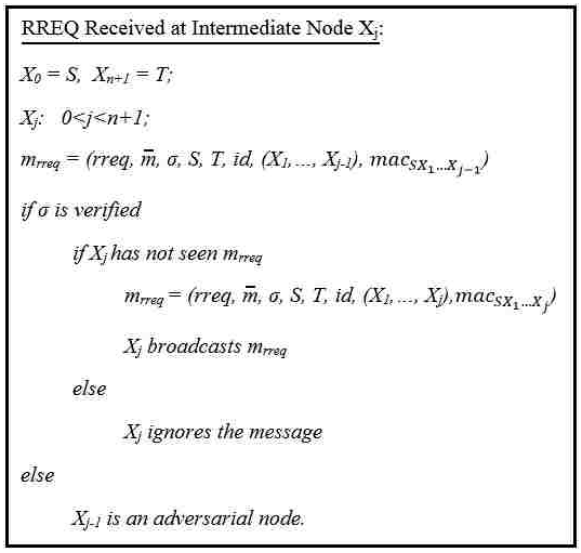
$T \longrightarrow C: (REPLY, \bar{m}', \sigma', S, T, id, (A, B, C))$

$C \longrightarrow B: (REPLY, \bar{m}', \sigma', S, T, id, (A, B, C))$

$B \longrightarrow A: (REPLY, \bar{m}', \sigma', S, T, id, (A, B, C))$

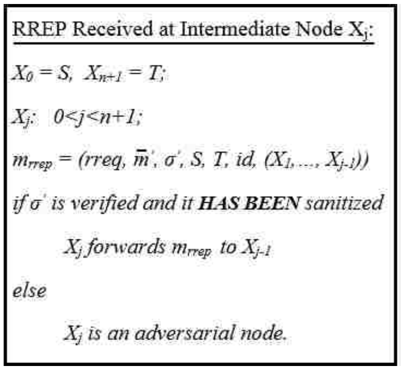
$A \longrightarrow S: (REPLY, \bar{m}', \sigma', S, T, id, (A, B, C))$

Figure 5.6: **An Illustration of Route Discovery Under our Protocol. The source S tries to find a route to the target node T.**



```
RREQ Received at Intermediate Node $X_j$:
$X_0 = S,\ X_{n+1} = T;$
$X_j$:  $0<j<n+1;$
$m_{rreq} = (rreq, \bar{m}, \sigma, S, T, id, (X_1, ..., X_{j-1}), mac_{SX_1...X_{j-1}})$
if $\sigma$ is verified
    if $X_j$ has not seen $m_{rreq}$
        $m_{rreq} = (rreq, \bar{m}, \sigma, S, T, id, (X_1, ..., X_j), mac_{SX_1...X_j})$
        $X_j$ broadcasts $m_{rreq}$
    else
        $X_j$ ignores the message
else
    $X_{j-1}$ is an adversarial node.
```

Figure 5.7: **Handling of Route Request by an Intermediate Node X_j.**



```
RREP Received at Intermediate Node $X_j$:
$X_0 = S,\ X_{n+1} = T;$
$X_j$:  $0<j<n+1;$
$m_{rrep} = (rreq, \bar{m}', \sigma', S, T, id, (X_1, \ldots, X_{j-1}))$
if $\sigma'$ is verified and it HAS BEEN sanitized
        $X_j$ forwards $m_{rrep}$ to $X_{j-1}$
else
        $X_j$ is an adversarial node.
```

Figure 5.8: **Handling of Route Reply by an Intermediate Node X_j**

5.5 Analysis and Discussion of the Attacks Detected by the Proposed Protocol

In this section, we discuss and show how our protocol prevents the hidden channel attacks on the protocols presented in Section 5.2.

5.5.1 Detecting and Preventing the Hidden Channel Attack on SRP

In SRP, intermediate nodes can easily inject invalid identifiers into route requests and forward them to a target T which in turn verifies these non-existent nodes correctly. This is possible because intermediate nodes neither check the source's MAC nor they append their own MACs to the route request. This kind of attack can be detected by our protocol. Consider the attack explained in Section 5.2.1, when the adversay A inserts the arbitrary sequence of identifiesr λ, under our protocol, the node list cannot pass the verification done by the target. Therefore, our protocol prevents such attacks.

5.5.2 Detecting and Preventing the attack on Ariadne with Signature

In this case, the adversarial node removes a node from the *Node List* (i.e., the route) such that the target can still validate the route (see Section 5.2.2). In Ariadne with Signature (see Section 2.5.3), each node appends the signature generated over the fields of the received route request. This gives ability to adversarial nodes (e.g., node A in Figure 5.2) to reconstruct the signatures when they receive multiple requests before rebroadcasting any of them. This signature technique leads to information leakage (attack) like the one presented in [4] as explained in Section 5.2.2. However, this attack is prevented by our protocol. When the iterating MAC is used, the adversarial node cannot reconstruct the MAC using multiple requests. Let's go back to Figure 5.2, where S initiates a route request to discover route to T, A receives the following two messages from nodes V and X respectively under Ariadne

$$msg_1 = (rreq, \bar{m}, \sigma, S, T, id, (Q, V), mac_{SQV}), \text{ and}$$

$$msg_2 = (rreq, \bar{m}, \sigma, S, T, id, (Q, V, W, X), mac_{SQVWX}).$$

Now, from message msg_2 A knows that V and W are neighbors, and A needs to create a message like this

$$msg_3 = (rreq, \bar{m}, \sigma, S, T, id, (Q, V, W, A), mac_{SQVWA}).$$

A needs to broadcast this message so that the target can validate all the intermediate nodes with their corresponding keys. However, in order to do that (i.e., to remove X from the route), A needs to get the iterated MAC calculated by W (i.e., mac_{SQVW}) which is protected by W's key. Therefore, A cannot remove X unless it knows the secrete key of W to calculate W's MAC (i.e., mac_{QVW}) and as a result, msg_3 cannot be created. Thus, our protocol prevents hidden channel attacks.

5.5.3 Detecting and Preventing the Attack on Basic Ariadne with MAC

Adversarial nodes in this scheme remove a node or a sequence of nodes located between two of the adversarial nodes as explained in Section 5.2.3. Intermediate nodes in Ariadne with MAC append their calculated MAC values to a list of MACs giving adversaries enough spaces to shorten the real route between source and target nodes. Consider the attack presented in Section 5.2.3, when the request arrives at the second adversarial node Y (see Figure 5.3), it drops the sequence (B, M, L, C) from the request.

Now, let's assume that the adversarial node can successfully remove the sequence mentioned above from the route request and the request arrives at T. Now, under our protocol, this attack is detected as follows. The target T puts the node list in the sanitizable position of the sanitizable signature which in turn can be verified by each intermediate node. The accumulated route that is sanitized in this example is (A, Y, D) after Y has dropped the intermediate nodes (B, M, L, C). Y cannot re-insert the hidden nodes (i.e., (B, M, L, C)) into the route reply received from D because each of these nodes can verify the sanitized signature which was computed by T on the received path (i.e., node list). If Y does so, the first node C in the node list will detect Y as an adversarial node because the sanitized signature will not be successfully verified. This prevents the reply from arriving at the source and hence, non-plausible routes are not created. Thus, our protocol prevents this type of hidden channel attacks.

5.5.4 Detecting and Preventing the attack on the Optimized Version of Ariadne with Iterated MAC

The iterating MAC calculated at each intermediate node is the most efficient scheme among the other flavors of Ariadne to prevent attacks. However, this optimized version is also prone to hidden channel attacks as discovered in [92] and presented in Section 5.2.4. This is because under this version of Ariadne, intermediate nodes check neither iterated MACs calculated by intermediate nodes nor the MAC computed by the target T.

In our protocol every node verifies that the signature is sanitized, hence, when a route reply arrives at intermediate nodes, it will be forwarded only if the verification of the sanitized signature succeeds.

Now, let's explain how our method prevents the hidden channel attack. Consider the hidden channel attack against Ariadne presented in Section 5.2.4; we notice that the adversarial node Y (see Figure 5.4) needs the iterating MAC (mac_{SAX}) that was broadcast by the other adversarial node X. But since there is an extra channel, B, between X and Y, X needs to impersonate Y and unicast a fake reply to B such that when B forwards the reply to X, Y can intercept the reply and extract the MAC generated by X (i.e., mac_{SAX}).

Now when X creates a fake route reply in the name of Y and unicasts it to B [92] (see Section 5.2.4), B checks the signature and finds it is not sanitized because T has not received the route request yet, hence B detects that Y is an adversarial node. Thus, our protocol prevents hidden channel attacks present in optimized Ariadne with *iterated MAC.*

Then, after the reply is sanitized by T, the route reply cannot be tampered with by an intermediate node. The source or the next upstream/downstream intermediate node to the adversarial node can detect the modification, hence the route reply arrives intact at the source S.

It is noteworthy to mention that we use one signature that is signed by the source S and sanitized by the target T. This signature is verified by the intermediate nodes in the request phase as well as by those in the reply phase. This important feature makes the proposed routing protocol securely propagate route requests towards target nodes (downstream flow) and also securely forward route replies towards source nodes (upstream flow) at a cost of one signature which is more cost effective than other schemes in which each node uses its own individual signature.

5.5.5 Detecting and Preventing the attack on endairA

The *endairA* protocol secures route replies by digitally signing but does not secure route requests. Therefore, adversarial nodes need a strategy to hide control information while route replies are forwarded to source nodes. Therefore, the attack found in [92] on *endariA* presented in Section 5.2.5 was based on this strategy. The adversarial node Y (Figure 5.5) needs to send the signature list (i.e., (sig_T, sig_D, sig_Y)) to the other adversarial node X so that the source S accepts the reply. It is assumed that there has been a route discovery initiated by node D towards the target A prior to the route discovery from S to T. Y uses this previous route discovery to hide the signature list in the *id* of the route request so that X can reconstruct the required information (i.e., (sig_T, sig_D, sig_Y)) from the modified id. The main reason for this attack is that route requests are not secured.

This attack is prevented by our protocol because the main part m (Section 5.4.3) of route requests in our protocol is authenticated, where

$$m = (id, m_{ID}, S, T, sanitizable - portion).$$

As explained in Section 5.4.3, m, which is an intrinsic part of the request, is signed by the source S using the sanitizable signature. Each intermediate node verifies this sanitizable signature, which is part of the request, before forwarding the request.

Now, to detect the attack on endairA, we go back to the scenario mentioned in Section 5.2.5 and briefly explained above. When the second adversarial node Y modifies id' of the request into some other identifier id'' to contain the signatue list, the main portion of the route request gets modified. This modification yields a signature different from the one sent the route request; hence, when the neighbor nodes of Y (i.e., intermediate nodes) receive this request, they discard it because the signature cannot be verified. Therefore, the modified request cannot travel any further beyond the neighbors of Y which means Y cannot communicate with X using invalid request information. As a result, this verification prevents such attacks which in turn means that our protocol prevents hidden channel attack present in endairA.

Chapter 6 Summary and Conclusion

In this chapter, we present a summary and conclusion of the protocols presented in this dissertation.

Many of the geographic routing protocols designed for MANETs either totally or partially depend on the idea of greedy forwarding in which packets are forwarded in the direction of the destination node. A neighbor node that is closer to the destination than the current node is picked as the next hop of the packet. However, this technique cannot always guarantee packet delivery since nodes may be non-uniformly distributed in the network resulting in dead ends or voids. Dealing with voids in the network is an important factor when designing a geographic routing protocol. Many protocols uses the right hand rule (face routing) as a recovery procedure to cope with that problem. Face routing provides a notable remedy to dead end problems but at the cost of building a planar graph of the network of local neighbors at each node and the cost of slightly increased end-to-end delay. Also, there are situations where, under face routing, packets cannot go around voids. We proposed a geographic protocol called GRB that solves this problem.

In Chapter 3, we presented GRB, a simple low-overhead position-based routing protocol which consistently and successfully delivers high percentage of data packets. We compared the performance of GRB with well known position-based protocol GPSR, the on-demand routing protocol AODV, and the Dynamic Source Routing (DSR) protocol. Our performance evaluation shows that GRB performs as good as

GPSR (with low overhead) and better than AODV and DSR under most scenarios. Unlike GPSR, GRB does not need to construct planar graphs to route around voids; it simply picks the best next hop to forward the data packets; hence GRB is simple. On the other hand, it achieves comparable packet delivery ratio to GPSR and AODV; hence it is robust.

We note from the detailed description of the protocol (see Section 3.2) that data packets are forwarded greedily. When greedy forwarding fails, GRB picks the next best hop based on simple heuristics without incurring large computation overhead, unlike GPRS. A packet can come back (backtrack) to its sender/forwarder if the next hop picked for forwarding packet could not use any of its neighbors to forward the packet. This feature makes GRB solve the dead end problem while, in some situations, GPSR fails.

In chapter 4, we presented HGRB, a simple low-overhead hybrid on-demand routing protocol that combines features of geographic routing protocols and topology based routing. HGRB consistently and successfully delivers high percentage of data packets at lower routing control overhead. We compared the performance of HGRB with the well-known on-demand routing protocol AODV and DSR. Our performance evaluation shows that HGRB performs better than AODV and DSR with respect to all metrics.

We evaluated the performance of HGRB in different terrain areas within which nodes move with different node density and node mobility patterns. The probability of a route breaking increases as the routes grow longer in a highly mobile environment. HGRB's packet delivery ratio is higher than AODV and DSR at all mobility modes on larger networks because HGRB uses only local topology information to forward RREQ packets; hence no penalty for HGRB as the path length increases from source nodes to destination nodes. To study the number of hops on established routes, we calculated the average hop count on the routes of all the received packets under all the flows for both HGRB and AODV. In small areas, the number of hops on routes established by AODV are comparable to that of HGRB; however, HGRB uses significantly less hop counts when network density increases because chances for RREQ packets to propagate through greedy paths is higher which in turn reduces the hop count. The routing control overhead for AODV and DSR is much higher than HGRB especially in high mobility environments (i.e., lower pause times) because routes break more frequently and that costs higher volumes of RREQ packets under both AODV and DSR. However, since RREQ packet forwarding decisions are made locally in HGRB, no additional cost is incurred for path discovery in low and high mobilities which makes HGRB superior to both AODV and DSR in routing control overhead.

In chapter 5, we discussed how some of the secure routing protocols proposed for MANETs are in fact not secure, and presented a novel, secure routing protocol for MANETs called SAriadne. SAriadne relies on sanitizable signature, a scheme that allows other parties to sanitize the signature so that the original signer and other intermediate nodes can validate the sanitized signature. It detects and prevents hidden channel attacks for MANETs. The security mechanism we designed is general and can be applied to any source routing protocol. A comprehensive analysis of the hidden channel attacks on SRP, Ariadne, and endairA is provided and we showed how our protocol detects and prevents these hidden channel attacks.

Bibliography

[1] A. Maghsoudlou, M. St-Hilaire, and T. Kunz, "A survey on geographic routing protocols for mobile ad hoc networks," *Systems and Computer Engineering, Technical Report SCE-11-03.–Carleton University.–2011.–49 p.*

[2] Y. Zhao, Y. Chen, B. Li, and Q. Zhang, "Hop id: A virtual coordinate based routing for sparse mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 9, pp. 1075–1089, Sept 2007.

[3] G. Acs, L. Buttyan, and I. Vajda, "Provably secure on-demand source routing in mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 11, pp. 1533–1546, 2006.

[4] L. Buttyán and I. Vajda, "Towards provable security for ad hoc routing protocols," in *Proceedings of the 2Nd ACM Workshop on Security of Ad Hoc and Sensor Networks.* New York, NY, USA: ACM, 2004.

[5] B. A. Mahmood and D. Manivannan, "Position based and hybrid routing protocols for mobile ad hoc networks: a survey," *Wireless Personal Communications*, vol. 83, no. 2, pp. 1009–1033, 2015.

[6] J. Lin and G.-S. Kuo, "A novel location-fault-tolerant geographic routing scheme for wireless ad hoc networks," in *Proceedings of 63rd Vehicular Technology Conference*, May 2006.

[7] R. Flury and R. Wattenhofer, "Randomized 3d geographic routing," in *Proceedings of 27th IEEE international Conference on Computer Communications (INFOCOM)*, April 2008.

[8] B. Karp and H. T. Kung, "Gpsr: Greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, 2000.

[9] H. Takagi and L. Kleinrock, "Optimal transmission ranges for randomly distributed packet radio terminals," *IEEE Transactions on communications*, vol. 32, no. 3, pp. 246–257, 1984.

[10] T.-C. Hou and V. Li, "Transmission range control in multihop packet radio networks," *IEEE Transactions on Communications*, vol. 34, no. 1, pp. 38–44, 1986.

[11] M. Mauve, J. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad hoc networks," *IEEE network*, vol. 15, no. 6, pp. 30–39, 2001.

[12] Z. J. Haas, "A new routing protocol for the reconfigurable wireless networks," in *Proceedings of IEEE 6th International Conference on Universal Personal Communications Record*, Oct 1997.

[13] C. E. Perkins and P. Bhagwat, “Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers,” *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 4, pp. 234–244, Oct. 1994.

[14] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward, “A distance routing effect algorithm for mobility (dream),” in *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 1998.

[15] C. E. Perkins and E. M. Royer, “Ad-hoc on-demand distance vector routing,” in *Proceedings of Second IEEE Workshop on Mobile Computing Systems and Applications, WMCSA*, Feb 1999.

[16] H. Li and M. Singhal, “An anchor-based routing protocol with cell id management system for ad hoc networks,” in *Proceedings of International Conference on Computer Communications and Networks*, Oct 2005.

[17] F. Cadger, K. Curran, J. Santos, and S. Moffett, “A survey of geographical routing in wireless ad-hoc networks,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 621–653, 2013.

[18] C. Lemmon, S. M. Lui, and I. Lee, “Geographic forwarding and routing for ad-hoc wireless network: A survey,” in *Proceedings of Fifth International Joint Conference on INC, IMS and IDC.* IEEE, 2009.

[19] V. C. Giruka and M. Singhal, “A self-healing on-demand geographic path routing protocol for mobile ad-hoc networks,” *Ad Hoc Netw.*, vol. 5, no. 7, pp. 1113–1128, Sep. 2007.

[20] H. Y.-C. and A. Perrig, “A survey of secure wireless ad hoc routing,” *IEEE Security Privacy*, vol. 2, no. 3, pp. 28–39, May 2004.

[21] Y.-C. Hu, A. Perrig, and D. B. Johnson, “Ariadne: A secure on-demand routing protocol for ad hoc networks,” *Wireless networks*, vol. 11, no. 1-2, pp. 21–38, 2005.

[22] S. Marti, T. J. Giuli, K. Lai, and M. Baker, “Mitigating routing misbehavior in mobile ad hoc networks,” in *Proceedings of the 6th annual international conference on Mobile computing and networking.* ACM, 2000.

[23] H. Shen and L. Zhao, “Alert: An anonymous location-based efficient routing protocol in manets,” *IEEE Transactions on Mobile Computing*, vol. 12, no. 6, pp. 1079–1093, June 2013.

[24] K. Kavitha, K. Selvakumar, T. Nithya, and S. Sathyabama, “Zone based multicast routing protocol for mobile ad hoc network,” in *Proceedings of International Conference on Emerging Trends in VLSI, Embedded System, Nano Electronics and Telecommunication System (ICEVENT)*, Jan 2013.

[25] S. Jain, A. Shastri, and B. K. Chaurasia, "Analysis and feasibility of reactive routing protocols with malicious nodes in manets," in *Proceedings of International Conference on Communication Systems and Network Technologies (CSNT)*, April 2013.

[26] M. H. Mamoun, "A new proactive routing algorithm for manet," *International Journal of Academic Research*, vol. 2, no. 2, p. 50, 2010.

[27] V. N. Talooki, H. Marques, and J. Rodriguez, "Energy efficient dynamic manet on-demand (e2dymo) routing protocol," in *Proceedings of International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, June 2013.

[28] K. R. Gabriel and R. R. Sokal, "A new statistical approach to geographic variation analysis," *Systematic Biology*, vol. 18, no. 3, pp. 259–278, 1969.

[29] G. T. Toussaint, "The relative neighbourhood graph of a finite planar set," *Pattern recognition*, vol. 12, no. 4, pp. 261–268, 1980.

[30] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, "On the pitfalls of geographic face routing," in *Proceedings of the Joint Workshop on Foundations of Mobile Computing*, ser. DIALM-POMC '05. New York, NY, USA: ACM, 2005.

[31] H. Frey and I. Stojmenovic, "On delivery guarantees of face and combined greedy-face routing in ad hoc and sensor networks," in *Proceedings of the 12th annual international conference on Mobile computing and networking.* ACM, 2006.

[32] B. A. Mahmood and D. Manivannan, "Grb: Greedy routing protocol with backtracking for mobile ad-hoc networks," in *Proceedings of 12th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), IEEE*, Oct 2015.

[33] D. B. Johnson and D. A. Maltz, *Mobile Computing.* Springer US, 1996, ch. Dynamic Source Routing in Ad Hoc Wireless Networks, pp. 153–181.

[34] B. A. Mahmood, A. Ibrahim, and D. Manivannan, "Hybrid on-demand greedy routing protocol with backtracking for mobile ad-hoc networks," in *Proceedings of 9th IFIP Wireless and Mobile Networking Conference (WMNC)*, July 2016.

[35] B. Mahmood, A. Ibrahim, and D. Manivannan, "Sariadne: A secure source routing protocol to prevent hidden-channel attacks," in *Proceedings of 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), IEEE, 2016.*

[36] F. Kuhn, R. Wattenhofer, and A. Zollinger, "An algorithmic approach to geographic routing in ad hoc and sensor networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 16, no. 1, pp. 51–62, 2008.

[37] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," *Wireless networks*, vol. 7, no. 6, pp. 609–616, 2001.

[38] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Worst-case optimal and average-case efficient geometric ad-hoc routing," in *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing.* New York, NY, USA: ACM, 2003.

[39] E. Schiller, P. Starzetz, F. Rousseau, and A. Duda, "Binary waypoint geographical routing in wireless mesh networks," in *Proceedings of the 11th international symposium on Modeling, analysis and simulation of wireless and mobile systems.* ACM, 2008.

[40] J. Na and C.-k. Kim, "Glr: A novel geographic routing scheme for large wireless ad hoc networks," *Computer networks*, vol. 50, no. 17, pp. 3434–3448, 2006.

[41] J. Na, D. Soroker, and C.-k. Kim, "Greedy geographic routing using dynamic potential field for wireless ad hoc networks," *IEEE Communications Letters*, vol. 11, no. 3, pp. 243–245, 2007.

[42] B. Leong, B. Liskov, and R. Morris, "Geographic routing without planarization," in *Proceedings of NSDI*, 2006.

[43] S. Rührup and I. Stojmenovi, "Contention-based georouting with guaranteed delivery, minimal communication overhead, and shorter paths in wireless sensor networks," in *Proceedings of International Symposium on Parallel & Distributed Processing (IPDPS).* IEEE, 2010.

[44] L. Barriere, P. Fraigniaud, L. Narayanan, and J. Opatrny, "Robust position-based routing in wireless ad hoc networks with irregular transmission ranges," *Wireless Communications and Mobile Computing*, vol. 3, no. 2, pp. 141–153, 2003.

[45] A. E. Abdallah, T. Fevens, and J. Opatrny, "High delivery rate position-based routing algorithms for 3d ad hoc networks," *Computer Communications*, vol. 31, no. 4, pp. 807–817, 2008.

[46] P. Papadimitratos and Z. Haas, "Secure routing for mobile ad hoc networks," *MOBILE COMPUTING AND COMMUNICATIONS REVIEW*, vol. 1, no. 2, pp. 27–31, 2002.

[47] L. Zhao and H. Shen, "A low-cost anonymous routing protocol in manets," in *Proceedings of 18th Internatonal Conference on Computer Communications and Networks.* IEEE, 2009.

[48] G. Ghinita, M. Azarmi, and E. Bertino, "Privacy-aware location-aided routing in mobile ad hoc networks," in *Proceedings of Eleventh International Conference on Mobile Data Management.* IEEE, 2010.

[49] S. Kwon and N. B. Shroff, “Geographic routing in the presence of location errors,” *Computer Networks*, vol. 50, no. 15, pp. 2902–2917, 2006.

[50] A. Ali, L. A. Latiff, and N. Fisal, “Gps-free indoor location tracking in mobile ad hoc network (manet) using rssi,” in *Proceedings of RF and Microwave Conference*. IEEE, 2004.

[51] J. Lessmann, M. Schoeller, and F. Zdarsky, “Rope ladder routing: Position-based multipath routing for wireless mesh networks,” in *Proceedings of International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2010.

[52] I. Stojmenovic, “Localized network layer protocols in wireless sensor networks based on optimizing cost over progress ratio,” *IEEE Network*, vol. 20, no. 1, pp. 21–27, Jan 2006.

[53] W. Feng, L. Zhang, and J. M. Elmirghani, “Energy saving geographic routing in ad hoc wireless networks,” *IET communications*, vol. 6, no. 1, pp. 116–124, 2012.

[54] P. Sinha, *QoS Issues in Ad-Hoc Networks*. Boston, MA: Springer US, 2005, pp. 229–247.

[55] C. Huang, F. Dai, and J. Wu, “On-demand location-aided qos routing in ad hoc networks,” in *Proceedings of International Conference on Parallel Processing*. IEEE, 2004.

[56] E. Kranakis, H. Singh, and J. Urrutia, “Compass routing on geometric networks,” in *Proceedings of International Conference on 11th Canadian Conference on Computational Geometry*, 1999.

[57] B. Leong, S. Mitra, and B. Liskov, “Path vector face routing: Geographic routing with local face information,” in *Proceedings of 13th International Conference on Network Protocols*. IEEE, 2005.

[58] M. Heissenbüttel, T. Braun, T. Bernoulli, and M. WäLchli, “Blr: beaconless routing algorithm for mobile ad hoc networks,” *Computer communications*, vol. 27, no. 11, pp. 1076–1086, 2004.

[59] Y. Xu, W.-C. Lee, J. Xu, and G. Mitchell, “Psgr: priority-based stateless georouting in wireless sensor networks,” in *Proceedings of International Conference on Mobile Adhoc and Sensor Systems*. IEEE, 2005.

[60] Q. Fang, J. Gao, and L. J. Guibas, “Locating and bypassing routing holes in sensor networks,” in *Proceedings of INFOCOM. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, March 2004.

[61] R. Jain, A. Puri, and R. Sengupta, “Geographical routing using partial information for wireless ad hoc networks,” *IEEE Personal communications*, vol. 8, no. 1, pp. 48–57, 2001.

[62] D. Chen and P. K. Varshney, “On-demand geographic forwarding for data delivery in wireless sensor networks,” *Computer Communications*, vol. 30, no. 14, pp. 2954–2967, 2007.

[63] F. Ye, A. Chen, S. Lu, and L. Zhang, “A scalable solution to minimum cost forwarding in large sensor networks,” in *Proceedings of 10th International Conference on Computer Communications and Networks*. IEEE, 2001.

[64] L. Zou, M. Lu, and Z. Xiong, “Pager-m: A novel location-based routing protocol for mobile sensor networks,” in *Proceedings of Broadwise*, October 2004.

[65] S. Chen, G. Fan, and J.-H. Cui, “Avoid ’void’ in geographic routing for data aggregation in sensor networks,” *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 1, no. 4, pp. 169–178, 2006.

[66] D. S. De Couto and R. Morris, “Location proxies and intermediate node forwarding for practical geographic forwarding,” 2001.

[67] L. Blažević, S. Giordano, and J.-Y. Le Boudec, “Self organized terminode routing,” *Cluster Computing*, vol. 5, no. 2, pp. 205–218, 2002.

[68] J. Li, J. Jannotti, D. S. J. D. Couto, D. R. Karger, and R. Morris, “A scalable location service for geographic ad hoc routing,” in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*. ACM, 2000.

[69] M. Mauve, J. Widmer, and H. Hartenstein, “A survey on position-based routing in mobile ad hoc networks,” *IEEE Network*, vol. 15, no. 6, pp. 30–39, Nov 2001.

[70] Y.-B. Ko and N. H. Vaidya, “Location-aided routing (lar) in mobile ad hoc networks,” *Wireless Networks*, vol. 6, no. 4, 2000.

[71] L. Blazevic, L. Buttyan, S. Capkun, S. Giordano, J. P. Hubaux, and J. Y. L. Boudec, “Self organization in mobile ad hoc networks: the approach of terminodes,” *IEEE Communications Magazine*, vol. 39, no. 6, pp. 166–174, Jun 2001.

[72] Z. J. Haas and B. Liang, “Ad hoc mobility management with uniform quorum systems,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 228–240, Apr 1999.

[73] R. Morris, J. Jannotti, F. Kaashoek, J. Li, and D. Decouto, “Carnet: A scalable ad hoc wireless network system,” in *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*. ACM, 2000.

[74] I. Stojmenovic, “Home agent based location update and destination search schemes in ad hoc wireless networks,” *Computer Science, SITE, University of Ottawa, TR-99-10*, 1999.

[75] X. Li, N. Mitton, A. Nayak, and I. Stojmenovic, "Localized load-aware geographic routing in wireless ad hoc networks," in *Proceedings of IEEE International Conference on Communications (ICC)*, June 2012.

[76] A. Macintosh, M. Ghavami, M. F. Siyau, and S. Ling, "Local area network dynamic (landy) routing protocol: A position based routing protocol for manet," in *Proceedings of 18th European Wireless Conference.* VDE, 2012.

[77] H. Li and M. Singhal, "Arpc: Anchor-based routing protocol for mobile ad hoc networks with cell id management system." *Adhoc & Sensor Wireless Networks*, vol. 4, no. 3, 2007.

[78] B. Zhou, Y.-Z. Lee, and M. Gerla, "Direction assisted geographic routing for mobile ad hoc networks," in *Proceedings of MILCOM Military Communications Conference.* IEEE, 2008.

[79] M. Gerla, Y.-Z. Lee, B. Zhou, J. Chen, and A. Caruso, "Direction forwarding for highly mobile, large scale ad hoc networks," in *Challenges in Ad Hoc Networking.* Springer, 2006, pp. 357–366.

[80] G. Pei, M. Gerla, and T.-W. Chen, "Fisheye state routing: a routing scheme for ad hoc wireless networks," in *Proceedings of IEEE International Conference on Communications*, 2000.

[81] M. Al-Rabayah and R. Malaney, "A new hybrid location-based ad hoc routing protocol," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM).* IEEE, 2010.

[82] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking.* ACM, 2003.

[83] H. Krawczyk, M. Bellare, and R. Canetti, "Hmac: Keyed-hashing for message authentication," United States, 1997 (RFC).

[84] A. Perrig, R. Canetti, J. Tygar, and D. Song, "Efficient authentication and signing of multicast streams over lossy channels," in *Proceedings of IEEE Symposium on Security and Privacy, S P*, 2000.

[85] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, 1978.

[86] Y. Hu, A. Perrig, and D. B. Johnson, "Efficient security mechanisms for routing protocols," in *Proceedings of NDSS*, 2003.

[87] M. Shobana and S. Karthik, "A performance analysis and comparison of various routing protocols in manet," in *Proceedings of International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME).* IEEE, 2013.

[88] X. Zeng, R. Bagrodia, and M. Gerla, "Glomosim: a library for parallel simulation of large-scale wireless networks," in *Proceedings of Twelfth Workshop on Parallel and Distributed Simulation*. IEEE, 1998.

[89] C. C. Hsu and C. L. Lei, "A geographic scheme with location update for ad hoc routing," in *Proceedings of Fourth International Conference on Systems and Networks Communications, ICSNC*, Sept 2009.

[90] A. K. Abdelaziz, M. Nafaa, and G. Salim, "Survey of routing attacks and countermeasures in mobile ad hoc networks," in *Proceedings of IEEE International Conference on Computer Modelling and Simulation (UKSim)*, April 2013.

[91] G. Ateniese, D. H. Chou, B. Medeiros, and G. Tsudik, *Computer Security: Proceedings of European Symposium on Research in Computer Security.* Springer Berlin Heidelberg, 2005, ch. Sanitizable Signatures, pp. 159–177.

[92] M. B. and B. de Medeiros, "On the security of route discovery in manets," *IEEE Transactions on Mobile Computing*, vol. 8, no. 9, pp. 1180–1188, Sept 2009.

[93] G. A. and Breno Medeiros, *Security in Communication Networks: International Conference, SCN.* Springer Berlin Heidelberg, 2004, ch. On the Key Exposure Problem in Chameleon Hashes, pp. 165–179.

[94] S. R. M. Krishna, P. V. K. Prasad, M. N. S. Ramanath, and B. M. Kumari, "Security in manet routing tables with fmnk cryptography model," in *Proceedings of International Conference on Electrical, Electronics, Signals, Communication and Optimization (EESCO)*, Jan 2015.

Vita

Name: Baban Ahmed Mahmood

Education:

- B.Sc. in Computer and Software Engineering, University of Al-Mustansryah, Baghdad, Iraq, 2003
- M.Sc. in Computer Science, University of Sulaimaniya, Kurdistan, Iraq, 2009.
- M.Sc. in Computer Science, University of Kentucky, 2015.

Employment:

- 2003-2006, Teaching Assistant, University of Kirkuk.
- 2009-2011, Instructor, University of Kirkuk.
- Spring 2012 - Fall 2016, 7 Semesters, Teaching Assistant, University of Kentucky.

Publications:

Journal Publications:

- B. Mahmood and D. Manivannan, “Position Based and Hybrid Routing Protocols for Mobile Ad Hoc Networks: a Survey,” in *Wireless Personal Communications, SPRINGER,* vol. 83, no. 2, pp. 1009-1033, January 2015.
- B. Mahmood and D. Manivannan, “Hybrid On-demand Greedy Routing Protocol with Backtracking for Mobile Ad-Hoc Network, submitted to *Pervasive and Mobile Computing, ELSEVIER.*
- B. Mahmood and D. Manivannan, “GRB: Greedy Routing Protocol with Backtracking for Mobile Ad-Hoc Network, submitted to *Wireless Networks, Springer*.
- B. Mahmood and D. Manivannan, “SAriadne: A Secure Source Routing Protocol to Prevent Hidden-Channel Attacks, to be submitted to *IEEE Transactions on Mobile Computing, IEEE.*

Conference Publications:

- B. Mahmood and D. Manivannan, “GRB: Greedy Routing Protocol with Backtracking for Mobile Ad-Hoc Network,” in *Proceedings of the 12th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, October 2015.

- B. Mahmood, A. Ibrahim, and D. Manivannan, "Hybrid On-demand Greedy Routing Protocol with Backtracking for Mobile Ad-Hoc Network," in *Proceedings of the IEEE 9th IFIP Wireless and Mobile Networking Conference (WMNC)*, July 2016.

- B. Mahmood, A. Ibrahim, and D. Manivannan, "SAriadne: A Secure Source Routing Protocol to Prevent Hidden-Channel Attacks," in *Proceedings of the 12th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, October 2016.

- A. Ibrahim, B. Mahmood, and M. Singhal, "A Secure Framework for Sharing Electronic Health Records over Clouds," in *Proceedings of the 4th IEEE International Conference on Serious Games and Applications for Health (SeGAH)*, May 2016.

- A. Ibrahim, B. Mahmood, and M. Singhal, "A Secure Framework For Medical Information Exchange (MI-X) Between Healthcare Providers," in *Proceedings of the 4th IEEE International Conference on Healthcare Informatics (ICHI)*, October 2016.