

A TECHNIQUE TO DETECT A BLACK HOLE IN AD HOC NETWORKS

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Sciences

By

Vijay Anand Suravarapu

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

November 2010

Fargo, North Dakota

North Dakota State University
Graduate School

Title

A TECHNIQUE TO DETECT A BLACK HOLE

IN AD HOC NETWORKS

By

VIJAYANAND SURAVARAPU

The Supervisory Committee certifies that this *disquisition* complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

North Dakota State University Libraries Addendum

To protect the privacy of individuals associated with the document, signatures have been removed from the digital version of this document.

ABSTRACT

Suravarapu, Vijay Anand, M.S., Department of Computer Science, College of Science and Mathematics, North Dakota State University, November 2010. A Technique to Detect a Black Hole in Ad Hoc Networks. Major Professor: Dr. Kendall Nygard.

Mobile Ad Hoc Networks (MANETs) are important because they are widely used in war zones, rescue missions and other civilian applications where it is almost impossible to use any other networks. MANETs are temporary networks where nodes are allowed to join or leave the network at any point in time. Though MANETs offer flexibility, self maintenance and dynamic topology, they suffer from limited resources, have less computational power and are susceptible to attacks. Due to the dynamic nature of MANETs, the secure transmission of data is one of the key issues in MANETs. Thus there is a need to safeguard the network from internal as well as external security threats.

In this paper, I attempt to improve the security environment of one of the most popular routing protocols in MANETs, the Ad Hoc On Demand Distance Vector Routing Protocol (AODV). AODV is a reactive protocol which guarantees loop free routes during the transmission of data, and the focus of this paper is mainly to secure the AODV protocol against black hole attacks. This paper proposes the Forward Reaching Black Hole Detection algorithm (FRBD) to detect single black holes in MANETs that are based on the AODV routing protocol. This paper presents a solution that includes slight modifications in the functioning of the AODV protocol. I demonstrate the proposed solution on a specially developed Java based simulation tool. The paper examines the black hole detection capabilities of the FRBD algorithm by applying it on randomly generated problem instances. These instances vary based on the total number of nodes, paths and the location

of the black hole node. Our analysis shows that the FRBD algorithm effectively detects single black holes in the network and thus improves the overall security of MANETs.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor, Dr. Kendall Nygard, for his assistance, support and guidance. Dr. Nygard's recommendations and suggestions have been invaluable for this work.

I would also like to thank Dr. Changhui Yan, Dr. Jun Kong and Dr. Limin Zhang for being a part of my Masters Paper committee.

Finally, words alone cannot express the thanks I owe to my family and friends for their continuous encouragement and support.

TABLE OF CONTENTS

ABSTRACT.....	iii
ACKNOWLEDGEMENTS.....	v
LIST OF TABLES.....	ix
LIST OF FIGURES	x
CHAPTER 1. INTRODUCTION.....	1
CHAPTER 2. BACKGROUND AND LITERATURE REVIEW	3
2.1. Advantages of Wireless Networks.....	3
2.2. Modes of Wireless Networks.....	3
2.2.1. Infrastructure Mode.....	4
2.2.2. MANET Mode.....	4
2.3. Routing Protocols in MANETs.....	5
2.3.1. Proactive (Table Driven) Routing Protocols	6
2.3.2. Reactive (On Demand) Routing Protocols.....	6
2.3.2.1. Overview of AODV Protocol	7
2.4. Security Issues in MANETs.....	9
2.4.1. Worm Hole Attacks.....	9
2.4.2. Gray Hole Attacks	10
2.4.3. Black Hole Attacks.....	10
2.5. Prevention of Black Hole Attacks (Literature Review).....	12

- CHAPTER 3. PROCEDURE 16
 - 3.1. Black Hole..... 16
 - 3.2. Modified AODV 16
 - 3.2.1. PATH Variable..... 17
 - 3.2.2. FORWARD NODE Variable 17
 - 3.2.3. RQUERY Control Message 18
 - 3.3. Generating Suspects List..... 19
 - 3.3.1. Suspects List Generation Algorithm 19
 - 3.3.2. Improving Efficiency of the Suspects List Generation Algorithm..... 20
 - 3.3.2.1. Set Coverage..... 20
 - 3.4. Forward Reaching Black Hole Detection Algorithm (FRBD)..... 22
 - 3.4.1. Inputs 22
 - 3.4.2. The Process..... 22
- CHAPTER 4. RESULTS..... 26
 - 4.1. Test Cases 28
 - 4.2. Analysis of Results..... 34
- CHAPTER 5. SIMULATION TOOL..... 36
 - 5.1. Development Environment 36
 - 5.2. Tool 36
 - 5.2.1. Inputs 36

5.2.2. Outputs	37
5.2.3. Software Domain Design	37
5.2.3.1. Basic Use Case Diagram.....	38
5.2.3.2. Class Diagram Overview	39
5.2.3.3. Class Diagram Detailed	40
5.2.3.4. Sequence Diagram	41
5.2.4. User Interface Design	41
5.2.4.1. Overview.....	41
5.2.4.2. Navigational Steps	42
5.3. AMPL Set Coverage Algorithm	44
5.3.1. Sample Problem.....	45
5.3.1.1. Input File.....	46
5.3.1.2. The Process	47
5.3.2. Interfacing AMPL with JAVA	50
CHAPTER 6. CONCLUSIONS AND FUTURE WORK.....	52
REFERENCES	53

LIST OF TABLES

<u>Table</u>	<u>Page</u>
1. Test Case # 1 (Multi Connected Black Hole, 2 Suspects, Black Hole Found).....	29
2. Test Case # 2 (Multi Connected Black Hole, 3 Suspects, Black Hole Found).....	29
3. Test Case # 3 (Isolated Leaf Black Hole, 4 Suspects, Black Hole Found).....	29
4. Test Case # 4 (Isolated Leaf Black Hole, 19 Suspects, Black Hole Found).....	30
5. Test Case # 5 (Multi Connected Black Hole, 13 Suspects, Black Hole Found).....	31
6. Test Case # 6 (Isolated Leaf Black Hole, 6 Suspects, Black Hole Found).....	31
7. Test Case # 7 (Isolated Black Hole, 17 Suspects, Black Hole Not Found).....	32
8. Test Case # 8 (Multi Connected Black Hole, 0 Suspect, Black Hole Not Found)	32
9. Test Case # 9 (Isolated Leaf Black Hole, 24 Suspects, Black Hole Found).....	33
10. Test Case # 10 (Isolated Black Hole, 15 Suspects, Black Hole Not Found).....	34

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
1. Two Devices Communicating Through Infrastructure based Network.....	4
2. Three Devices Communicating Through MANET Based Network.....	4
3. Classification of Routing Protocols in MANETs	5
4. A Sample MANET Consisting of a Black Hole	12
5. Pseudo Code – Generating Suspects List	19
6. Integer Linear Program Formulation for Set Coverage.....	21
7. AMPL Set Coverage Algorithm Sample Inputs and Outputs.....	22
8. Pseudo Code – Forward Reaching Black Hole Detection Algorithm	24
9. Forward Reaching Black Hole Detection Algorithm Block Diagram.....	25
10. Multi Connected Black Hole Node.....	26
11. Isolated Leaf Black Hole Node.....	27
12. Isolated Black Hole Node	27
13. Basic Use Case Diagram.....	38
14. Class Diagram Overview	39
15. Class Diagram Detailed	40
16. Sequence Diagram	41
17. Simulation Tool User Interface.....	42
18. Simulation Tool Input File to Generate Nodes Network	43
19. Results Output File	45
20. Sample Paths in Source's <i>in_generated_paths_list</i> buffer	45
21. AMPL Model File.....	46

22. AMPL Data File.....	47
23. Screenshot AMPL Command Line – Loading Model and Data Files.....	48
24. Screenshot AMPL Command Line-Solve Command.....	49
25. Screenshot AMPL Command Line – Display Command.....	49
26. AMPL Set Coverage Results	50
27. Setting up Connection Between AMPL and JAVA.....	51

CHAPTER 1. INTRODUCTION

There have been rapid advances in the development of wireless technology. Whether it is a call made through a cell phone or accessing Internet through PDAs, controlling satellites through base stations or broadcasting television signals, wireless technology is almost everywhere. The wide acceptance of wireless networks is due to its convenience: wireless networks carry advantages such as mobility, portability, low infrastructure cost and rapid connectivity.

Although wireless technology seems acceptable, there are some serious problems that need to be addressed in wireless networks. The secure transmission of data in wireless sensor networks has always been an issue of paramount concern, especially in Mobile Ad Hoc Networks (MANETs), where the dynamic topology provides the sensors with the ability to add or remove itself from the network without much effort, thus compromising its security. In addition, the lack of infrastructure support, limited power and wireless nature exposes MANET to various attacks. Steps need to be taken to safeguard the network from intruders and outside elements to maintain the integrity of data that is communicated between these devices.

The network layer of routing protocols in MANETs are the most susceptible to attacks as they rely on cooperative communication between nodes for finding routes, sending data packets etc. This paper proposes an algorithm to counter the black hole attacks against the AODV routing protocol. Black holes are malicious nodes that add themselves to the MANETs with the intent to eat up packets that are supposed to be delivered to the destination node. AODV is one of the most popular protocols proposed for use in MANETs. The proposed algorithm titled "Forward Reaching Black Hole Detection

(FRBD)” is effective against detecting single black holes in the network. We modified the underlying AODV protocol to suit our algorithmic needs, developed a simulation environment and processed a series of unique test cases to verify the described approach.

The rest of this paper is organized as follows. The next chapter briefly describes the basics of wireless networks, advantages, modes, AODV protocol and also different approaches proposed in literature for finding a black hole in AODV protocol. Chapter 3 focuses on the design aspects of our proposed algorithm. It also introduces the modified AODV protocol that has been slightly altered to suit the proposed algorithm. Chapters 4 present the experimental setup and the evaluation of results. Chapter 5 describes the simulation environment along with the tool design, inputs, outputs and other UI description. Chapter 6 concludes the paper and points to areas for future research.

CHAPTER 2. BACKGROUND AND LITERATURE REVIEW

Wireless networks are used to transmit data among users in an environment where there is an absence of wired infrastructure or where the nature of the connection needs to be temporary. The popularity of wireless networks has increased in the last decade due to the developments in the use of laptops, PDAs, cell phones etc. This chapter introduces the basics of wireless networks and includes some ideas from other literatures oriented towards the direction of black hole detection.

2.1. Advantages of Wireless Networks

Wireless networks are popular due to following reasons:

- *Convenience*: It is convenient for the end user to have wireless connectivity wherever they are without having to carry connectors, cables or any other devices.
- *Ease of Installation*: It becomes easy to install a wireless network as it eliminates the need to pull cables through walls and ceilings.
- *Scalability*: Wireless networks can be configured to a variety of network topologies. Configuration can easily be changed from a peer to peer network to a more complex infrastructure based wireless network where thousands of users access the network simultaneously.
- *Cost Effective*: Wireless networks are cheaper to install.

2.2. Modes of Wireless Networks

Wireless Networks are classified into Infrastructure based and MANET based modes.

Both these modes are briefly explained below:

2.2.1. Infrastructure Mode

Infrastructure Mode as the name suggests requires some kind of infrastructure for communication between wireless devices. Usually, the wireless devices are connected to a wired network through an interface called an “Access point”, through which all the wireless devices in the network communicate. Figure 1 is accesses from (D-Link).

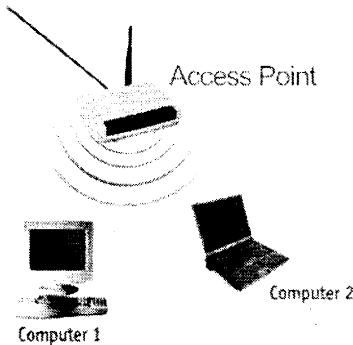


Figure 1. Two Devices Communicating Through Infrastructure based Network.

2.2.2. MANET Mode

Unlike infrastructure mode, the wireless devices in Ad hoc mode can communicate with each other directly. In this mode, wireless devices, which are within range of each other, are discovered and then connected directly in peer to peer fashion without requiring an access point. Figure 2 is accessed from (ADSL Gate).

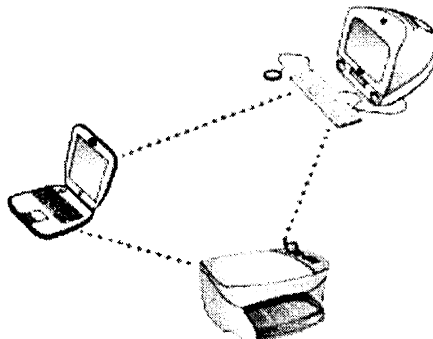


Figure 2. Three Devices Communicating Through MANET Based Network.

2.3. Routing Protocols in MANETs

Nodes in MANETs are highly mobile in nature, and as compared to a wired network, there are limitations in terms of available bandwidth, power consumption and high error rates. Also as compared to an infrastructure based network, nodes in MANETs cannot be connected in a random fashion. Nodes in a MANET have to participate in route discovery process and also have to maintain routing tables in order to communicate with each other. Based on these differences, routing protocols of wired networks cannot be applied to the MANETs, but there are numerous other protocols that have been developed for routing in MANET networks.

Routing protocols in MANETs are broadly classified into flat routing, hierarchical routing and GPS Routing (Refer to Figure 3). Flat routing protocols can be further classified into Proactive and Reactive protocols. Ad Hoc On Demand Vector (AODV) routing and Dynamic Source Routing (DSR) protocols are the two most reactive popular protocols used in MANETs (Ramaswamy, Fu, & Nygard, 2005).

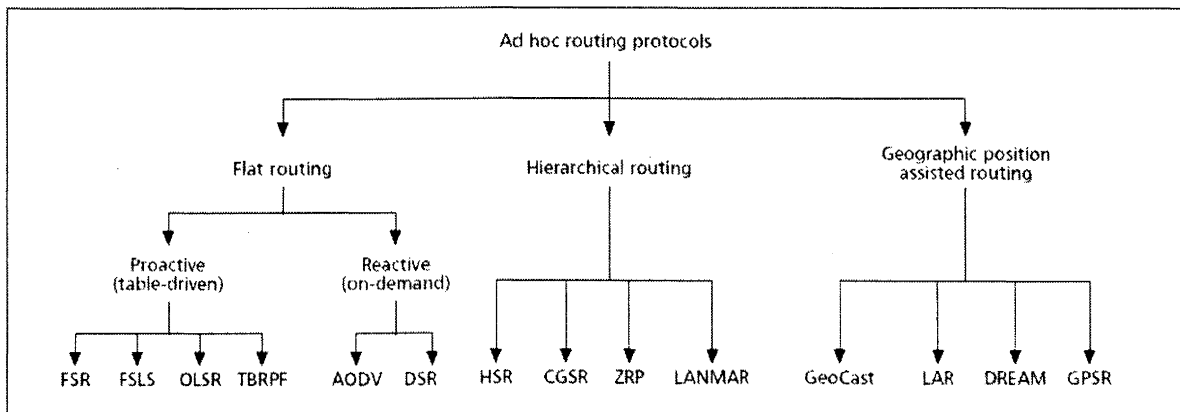


Figure 3. Classification of Routing Protocols in MANETs

This section describes the classification of Flat routing protocols, and AODV protocol falls in this category.

2.3.1. Proactive (Table Driven) Routing Protocols

In proactive routing, the nodes maintain information about the path from each node to every other node in the network. The main disadvantage of table driven routing protocols is that periodically updating the paths require extra overhead on the bandwidth. Also, periodically updating the routing table requires the nodes to be awake all the time, thus exhausting its battery power.

Examples of table driven routing protocols are Destination Sequenced Distance Vector Protocol (DSDV), Optimized Link State Routing (OLSR), Wireless Routing Protocol (WRP), Direction Forward Routing (DFR), Hierarchical State Routing Protocol (HSR), Distributed Bellman-Ford Routing Protocol (DBF) and Topology Dissemination based on Reverse-Path Forwarding Routing Protocol (TBRPF).

2.3.2. Reactive (On Demand) Routing Protocols

Reactive protocols are designed with the aim of reducing the control overhead, thus increasing bandwidth and conserving power at the nodes. These protocols maintain routes only to some specific routes for which the source has data to send, thus, the amount of bandwidth consumed is limited. Reactive protocols are source initiated protocols, and routes are established only when there is some data to be sent. The disadvantage of reactive protocols is that there is a latency associated with finding a path, and a network can be clogged due to excessive flooding of control and data packets.

Examples of reactive routing protocols are Ad hoc On Demand Distance Vector Routing Protocol (AODV), Robust Secure Routing Protocol (RSRP) and Admission Control enabled On-demand Routing (ACODV) and Cluster based Routing Protocol (CBRP).

Although all the different types of protocols used in MANETs is significant, our concentration is mainly focused around the AODV routing protocol. The description of protocol other than the AODV is out of scope for this paper.

2.3.2.1. Overview of AODV Protocol

Each mobile node in an AODV protocol acts as a router to forward information between source and destination nodes. Each node in the network maintains a routing table with routing information to reach its neighbors and also maintains two separate counters: a node sequence number and the broadcast id. The routing table contains the following fields:

- `destination_address`: Destination address of the node to which the current node previously maintained a path.
- `next_hop`: Next hop is used for reverse path setup (explained further in this section).
- `no_of_hops`:
- `destination_sequence_number`:

When a source node (say S) has to communicate with destination node (say D), it initiates the route discovery process by incrementing the broadcast id and broadcasting Route Request (RREQ) packet to its neighbors. The RREQ control packet contains the following fields:

- `source_address`: This field contains the address of S.
- `source_sequence_number`: This field maintains the freshness information about the route to the source.
- `destination_address`: This field contains the address of D.

- `destination_sequence_number`: This field specifies, how fresh a route should be before it is accepted by S.
- `hop_count`: This field is incremented by the intermediate nodes while processing the RREQ control packet.

The `<source_address, broadcast_id>` pair uniquely identifies a broadcast. As the RREQ control packet travels from node to node, a reverse path is automatically set from all these nodes back to the source. Each node that processes the RREQ packets notes the address of the node from which it received the RREQ in a process called a Reverse Path Setup.

If an intermediate node has the route to the desired destination in its routing table, it compares the `destination_sequence_number` in RREQ with that in the routing table. If the `destination_sequence_number` in the routing table is less than that in the RREQ, it rebroadcasts the RREQ to its neighbors. Otherwise, it unicasts a Route Reply (RREP) control packet to its neighbor from which it received the RREQ, which is done only if the same RREQ was not processed before the `<source_address, broadcast_id>` pair identifies this.

As the RREP travels back to S through the reverse route, each intermediate node along this path sets a forward pointer to the node from which it received the RREP and also records the `destination_sequence_number` to the requested destination in its routing table, a process called a Forward Path Setup. If the intermediate node receives another RREP after propagating the first one to S, it checks the `destination_sequence_number` of the new RREP. The intermediate node updates the entry to the requested destination in 2 cases-

- If the `destination_sequence_number` of the new RREP is greater than in the routing table, OR
- If the new `sequence_number` in the new RREP is the same as in the routing table and the `hop_count` is small.

If none of these cases are found to be true, the intermediate node drops the RREP, which guarantees loop free paths.

2.4. Security Issues in MANETs

MANETs are prone to security related adversaries due to their wireless nature and absence of any infrastructure support (Peng & Kun, 2003). Attacks on MANETs often target different layers of the routing protocol such as physical, data link and network layers, as these layers carry the most important functionality of MANETs. In this section, we will look into the various types of attacks on the network layer of the routing protocol.

Attacks on routing protocol's network layer generally have two motives: a malicious node does not forward packets (eats up packets) OR a malicious node changes routing message information such as sequence number, IP address etc. By attacking the network layer, the attackers can absorb the network traffic, inject themselves between the source and destination and thus control the network traffic flow (Wu, Chen, Wu, & Cardei, 2007).

There are numerous types of attacks oriented against the network layer, but for our convenience, we will only discuss three. Our concentration, however, will be on the black hole attacks.

2.4.1. Worm Hole Attacks

In a wormhole attack, a malicious node after receiving the control packets tunnels (reroutes) them to another malicious node, thus disrupting the routing process. This tunnel

between two colluding malicious nodes is referred to as a wormhole (Wu, Chen, Wu, & Cardei, 2007). Wormhole attacks can be severe if used against On Demand routing protocols such as AODV or DSV. This kind of control packet tunneling prevents the discovery of paths other than the ones through the wormhole.

2.4.2. Gray Hole Attacks

Gray Hole attacks are also called “routing misbehavior” attacks, in which a malicious node initially agrees to send the data to the destination, but fails to do so. When a source broadcasts a RREQ message, the malicious node sends a true RREP back to the source. After the source starts sending packets to this malicious node, it drops these packets without forwarding them to the destination, thus launching a “denial of service” attack.

When a malicious node’s neighbor tries to send data through the malicious node and loses connection to the destination due to the denial of the service attack launched by the malicious node, the neighboring node tries to find an alternative path to the destination by broadcasting the RREQ message. The malicious node replies to this RREQ with a true RREP message, thus taking control of the communication, but will launch another denial of service attack, prompting the neighboring node to again broadcast the RREQ message. This cycle of events repeats until the goal of the malicious node (such as exhausting the battery consumption, network resource consumption, etc) is achieved.

2.4.3. Black Hole Attacks

In a black hole attack, the malicious node exploits the MANET routing protocol by advertising itself as having a route to the destination although, in reality, it does not have any such valid route. When the source sends the data packets to the malicious node, it eats

up those packets without forwarding them to the intended destination, thus, the packets are lost.

The black hole (malicious) node waits for its neighboring nodes to send an RREQ message. When it receives an RREQ message from its neighbor, it responds with an RREP message and a very high destination sequence number and sends it back to the requester node. Requesting node checks for the destination sequence number and sends data to the path with a higher value of destination sequence number, at which point it begins to send data to the malicious node.

Consider a sample MANET network with 6 sensor nodes, one of which is a black hole node (Refer to Figure 4 below). We assume that all these nodes have similar sensing capabilities. Source (say S) intends to send some data packets to destination (D). Node S starts the route discovery process by broadcasting a RREQ message to all its neighbors (1, B and 4). Node 1, on receiving RREQ from S, rebroadcasts the RREQ to Node 2. Similarly, Node 4 rebroadcasts the RREQ to Node D. As Node B is a black hole, the only thing it does is reply back to Node S with a RREP and a very high destination sequence number. It is obvious that Node S receives more than one RREP, in this case a response from Node D (S-4-D) and a response from Node B (S-B). S compares the destination sequence of both these paths and finds the destination sequence number with node B to be of a higher value and so starts sending data to Node B. But as Node B is a black hole, it receives the data packets and drops them, an attack known as the “black hole attacks”.

Cryptographic techniques can be employed to protect against some of these attacks as it prevents intentional alterations of data. However, it can only be used as a preventive approach as it is incapable of dealing with situations such as selfish attacks or intentional

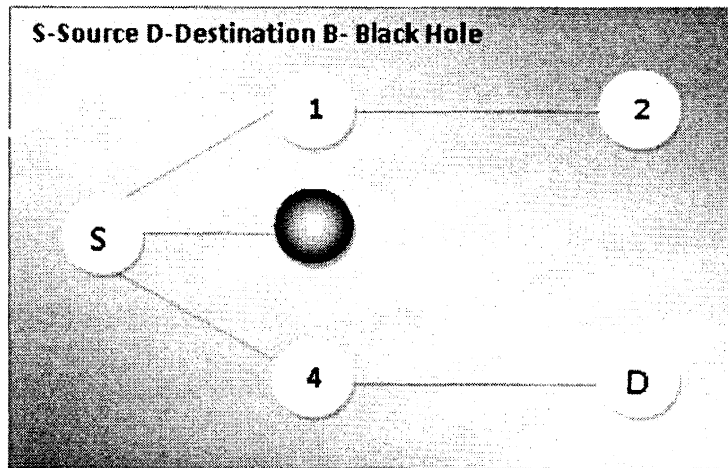


Figure 4. A Sample MANET Consisting of a Black Hole

dropping of packets by a malicious node, etc. There is a need to safeguard the wireless network with some kind of an intrusion detection system which ensures that the intruders are identified and removed from the network, thus preventing attacks.

2.5. Prevention of Black Hole Attacks (Literature Review)

The problem of black hole has been very severe in Ad-Hoc networks. Currently, there has been extensive research going on in the area of detecting black holes, and there are various solutions proposed in literature. The efficiency of these approaches might vary depending on the various factors such as the input parameters, simulation environment and more importantly, the logic used behind finding the black hole. The following section specifies some of the approaches described in the literature, which mentions some techniques that have served as a purposeful reference in developing the Forward Reaching Black Hole Detection (FRBD) algorithm.

Black Hole Attack in Mobile Ad Hoc Networks by Mohammad Al-Shurman, Seong-Moo Yoo and Seungjin Park (Al-Shurman, Yoo, & Park, 2004)

Solution1: The first solution described is to find more than one path to the destination. The source then unicasts a Route

Request (RREQ) message to the destination through these paths. All these messages must have different packet IDs and sequence numbers so that any intermediate node that processes these messages does not drop the second message if it has already processed the first one. The idea of this approach is for the source to wait for the RREP packet to arrive from multiple nodes, after which the source will extract the full paths to the destination and wait for more RREPs. The source buffers all the paths from the RREPs that it receives and tries to find a safe path to the destination out of these buffered paths. Two or more of these paths must have shared hops, from which the source can recognize the safe path to the destination. If there are no shared hops in these paths, then the source waits to receive another RREP. This solution guarantees to find a safe route to the destination, but time delay is the main drawback as the source has to receive and process many RREPs.

Solution2: Another solution has been proposed in the same publication (Al-Shurman, Yoo, & Park, 2004). Every packet in MANETs has a unique sequence number, which is an increasing value, (i.e., the next packet will have a greater sequence number, and so on). The intermediate node keeps the last processed sequence number from the same source to check if the packet was received before the same originating source. In this solution, every node needs to have two additional small sized tables, one to keep last packet sequence numbers for the last packet sent to every node and the other to keep last packet sequence numbers for the last packet received from every node. These tables are updated when any packet is processed by the node. Next, the source broadcasts the RREQ, and when the destination responds to the RREQ, the RREP contains the last packet sequence numbers received from this source. When an intermediate node with a path to the destination processes the RREQ, it will respond back to the source with an RREP that contains the last

packet sequence numbers received from the source by this intermediate node. This solution provides a reliable way to identify the suspicious node. There is no overlay added as the last packet sequence number is included with the control messages itself.

Improving AODV Protocol against Black hole Attacks by Nital Mistry, Devesh C Jinwala and Mukesh Zaveri (Mistry, Jinwala, & Zaveri, 2010). The proposed solution modifies only how the source node works by using an additional function. The source now stores all the RREPs in a table at its end instead of just accepting the first fresh enough RREP that comes to it. The source then analyzes all the RREPs received and discards the ones with a very high destination sequence number. The node that sends this RREP is suspected to be the black hole node, which is identified and then removed from the network.

Prevention of Cooperative Black Hole Attack in Wireless Ad Hoc Networks by Sanjay Ramaswamy, Huirong Fu, Manohar Sreekantharadhy, John Dixon and Kendall Nygard (Ramaswamy, Huirong, Manohar, Dixon, & Nygard, 2003). This approach slightly modifies the AODV protocol by introducing the concept of the Data Routing Information (DRI) table. Each node that responds to the RREQ message has to send two additional pieces (from and through bits) of information to the source. Each node maintains an additional DRI table, in which 1 stands for 'true' and 0 stands for 'false'. 'From' stands for the information on the routing data packet from the node, while 'through' stands for the information on the routing data packet through the node. In response to the RREQ from the source, the intermediate node (IN) has to provide its next hop node (NHN) and DRI entry for the NHN. If the source has used IN before for routing, then it is a reliable node; otherwise, additional checks are placed to find the identity of IN. This complete process is explained in the paper (Ramaswamy, Huirong, Manohar, Dixon, & Nygard, 2003). This

approach is effective in finding black holes in an environment where the malicious nodes cooperate with each other to avoid detection.

CHAPTER 3. PROCEDURE

To solve the issue of black hole detection, we propose the “Forward Reaching Black Hole Detection (FRBD)” algorithm. The RREP message format in the AODV protocol is slightly modified by introducing two extra fields called as PATH and FORWARD NODE. The source node after initiating the RREQ waits for some arbitrary amount of time to receive the RREPs back which are then buffered in a table at the source end. The source scans these paths and generates the list of suspects, which is passed as an input to the FRBD algorithm. The FRBD algorithm is applied over each suspect node which identifies if a suspect node is indeed a black hole or not. The efficiency of the black hole detection process is further improved by using the set coverage algorithm. The complete process is explained in this chapter.

3.1. Black Hole

Black Hole is “a node that does not have path to the destination but advertises itself as having one, with an intention of intercepting the packets which are intended for the destination”. These packets are intercepted and dropped by the black hole node without forwarding it to the destination, and the packets are lost as a result.

If an intermediate node is directly connected to the destination or has a path to the destination in its routing table, by definition, the node cannot be considered as a black hole as it has a valid path to the destination.

3.2. Modified AODV

The solution to detect the presence of black holes in the network involves modifying the RREP control packets as well as introducing a new control packet known as RQUERY

in the AODV protocol. The PATH and FORWARD NODE parameters are added to the existing RREP packets. The modified algorithm functions as follows. The source node (SN) broadcasts a RREQ to discover a safe route to the destination. As per the AODV protocol, an RREP can be generated either by a destination node (DN) or an intermediate node (IN) that has a fresh enough route entry for the destination node in its routing table. This RREP is unicast by the IN or DN to the neighboring node from which it receives the RREQ. An IN makes note of the neighboring from which it received the RREQ and then forwards the RREP in the reverse direction. When an RREP is generated by IN or DN, some additional information is also sent along with the RREP packet.

3.2.1. PATH Variable

In the modified AODV protocol, the PATH variable contains the absolute path information from SN to DN or IN (whoever responds to the RREQ). The IN or DN (RREP generator) adds its own node address to the PATH variable of RREP. This RREP is then unicast to the neighbor it received the RREQ from. The neighboring node processes this RREP by appending its own address to the PATH variable of RREP and forwards it to the neighboring node it received the RREQ from. This process repeats until the RREP reaches the SN. At this stage, the PATH variable of the RREP contains the complete path information from SN to IN or DN. For a single RREQ broadcasted, the SN may receive multiple RREPs from various nodes. The SN buffers all the PATH variable values in its buffer list.

3.2.2. FORWARD NODE Variable

The FORWARD NODE variable contains the “whom will I forward” value. While the PATH variable contains the complete path from IN or DN to SN, the FORWARD

NODE variable contains additional information for the SN to determine, to whom the IN is going to forward the data packets. It can only be set by the IN (that generates the RREP), as the DN does not forward the data to any other nodes. The FORWARD NODE value cannot be reset by any other IN that processes the RREP and is used for the black hole detection in the FRBD algorithm.

3.2.3. RQUERY Control Message

In the modified version of AODV protocol, we have introduced a new control message called Route Query (RQUERY). RQUERY is designed to request additional routing information from nodes during the black hole detection process in the FRBD algorithm. The RQUERY control message contains the following fields:

- *packet_id*: The packet id uniquely identifies a RQUERY control message sent by the SN for the destination.
- *path*: The SN specifies a complete path of how to reach the intended destination (ID). Once the RQUERY message reaches the ID and the required information is gathered, the ID uses the path variable value to forward the RQUERY message back to the SN.
- *suspect_node*: This field consists of the suspect node name. The purpose of RQUERY is to cross check the information that the suspect node provided about the “whom will I forward” node.
- *destination*: *The destination field of RQUERY consists of the destination node name for which the SN requested for a safe path to.*

- *Is_Suspect_Neighbour*: *Is_Suspect_Neighbour* is a Boolean true or false. The ID checks if the node in the Suspect Node field is its neighbor. If yes, then the *Is_Suspect_Neighbour* is true, otherwise, it is false.
- *Is_Path_to_Destination*: *Is_Path_to_Destination* is also a Boolean true or false. The ID checks if it has a valid route in its routing table to the node in the Destination field. If yes, then *Is_Path_to_Destination* is true, otherwise, it is false.

3.3. Generating Suspects List

3.3.1. Suspects List Generation Algorithm

The inputs to the FRBD algorithm are a list of suspect nodes generated by the SN by analyzing the paths stored in its buffer. The process of generating suspects is as follows:

An example path: {4, 2, 3, 6, 1}

4: IN or DN (rrep_generator_node),

1: SN.

The PATH value of each RREP that SN receives contains one such path. The SN buffers the PATH value and FORWARD NODE values of RREPs generated by DN in dn_generated_paths_list and the ones generated by IN in in_generated_paths_list. The list of suspects is generated by using the in_generated_paths_list and dn_generated_paths_list.

The algorithm for generating the list of suspects is as follows:

```

1 Get dn_generated_paths_list and in_generated_paths_list from SN
2 FOREACH path in in_generated_paths_list
3   Get rrep_generator_node
4   IF rrep_generator_node NOT IN suspects_list and dn_generated_paths_list
5     add rrep_generator_node to suspects_list
6   END IF
7 END FOR

```

Figure 5. Pseudo Code – Generating Suspects List

The characteristic of a black node is that it never forwards the RREQ, instead, it responds by generating RREP and sending it back to SN, so it is only the `in_generated_paths_list` that might contain the black hole generate path. For the very same reason, we can also conclude that the `dn_generated_paths_list` is guaranteed to be free of black holes.

In the “generating suspects list” algorithm, the SN loops through the paths in the `in_generated_paths_list`, and at every iteration, it finds the `rrep_generator_node`. If the `rrep_generator_node` is present in the `dn_generated_paths_list`, it cannot be a black hole, otherwise, SN adds the `rrep_generator_node` to the `suspects_list`. In this way, the suspects list is generated.

3.3.2. Improving Efficiency of the Suspects List Generation Algorithm

The list of suspects is an input to the FRBD algorithm. It is obvious that the less number of suspects, the faster FRBD algorithm is. In order to improve the efficiency of the overall black hole detection process, it is important to keep the suspects list as small as possible.

In this section, the set coverage algorithm is introduced, and this study’s goal is to minimize the number of paths in the `in_generated_paths_list` so that the “generating suspects list” algorithm becomes efficient, because it has to process a fewer number of paths and the number of suspects generated in the `suspects_list` is small. The details of the set coverage algorithm are as follows:

3.3.2.1. Set Coverage

The inputs to this algorithm are the sets of paths from the `in_generated_paths_list`, which may have some elements in common. The set coverage problem selects the

minimum number of these input sets so that these sets contain all the elements that are contained in any of the sets in the inputs.

The Integer Linear Program Formulation for Set coverage Algorithm is as follows.

```
set OBJECTS;
param npaths integer >= 0;
set COLS := 1..npaths;
param A {OBJECTS, COLS} >= 0 integer;
param C {COLS} >= 0, default 0 0;
param r integer >= 0;
param D {OBJECTS} >= 0;
var x {i in COLS} >= 0 integer;
minimize cost: sum {i in COLS} C[i] * x[i];
subject to setcover {i in OBJECTS}:
sum {j in COLS} A[i,j] * x[j] >= 1;
```

Figure 6. Integer Linear Program Formulation for Set Coverage

These types of linear program formulations can be solved using modeling language such as A Modeling Language for Mathematical Programming (AMPL). AMPL is a comprehensive and powerful algebraic modeling language for linear and nonlinear optimization problems, in discrete or continuous variables. We have used a MINOS solver, the default solver with the AMPL program.

The inputs to the AMPL program are the data file (.dat) and model file (.mod). Model file consists of the linear program formulation where as the data file consists of the paths from the `in_generated_paths_list`. Please refer to Chapter 5: Tool for a complete process description of loading input files, execution and generating results in AMPL.

Sample input and output data for the Set Coverage Integer Linear Program Formulation are shown in Figure 7. As we can see in Figure 7, there were 10 paths in the `in_generated_paths_list` and by using the AMPL Set Coverage Algorithm, the number of

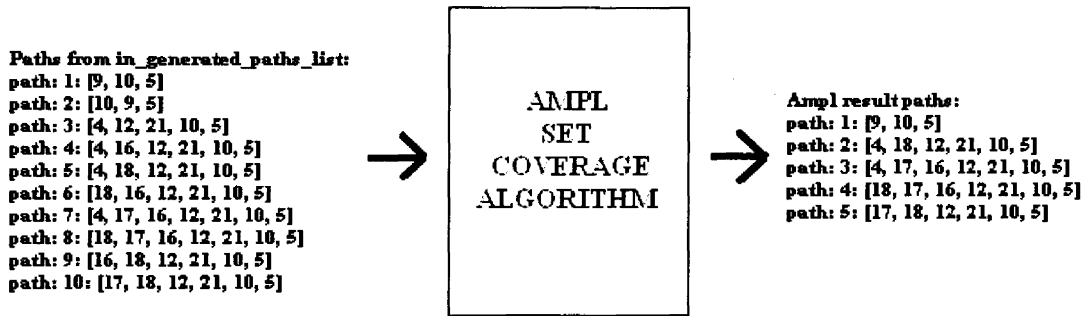


Figure 7. AMPL Set Coverage Algorithm Sample Inputs and Outputs

paths was optimized to 5. Instead of suspects being {9, 10, 4, 18, 16, 17}, the new suspects_list are {9, 4, 18, 17}. Therefore, the FRBD algorithm has to process just 4 suspect elements instead of 6, thus improving the overall efficiency of the black hole detection process.

3.4. Forward Reaching Black Hole Detection Algorithm (FRBD)

3.4.1. Inputs

The inputs to the FRBD algorithm are the set of suspect nodes generated by the SN (review section 3.3).

3.4.2. The Process

The process of finding black holes using the FRBD algorithm involves various steps such as looping through suspects list, sending an RQUERY message, reading RQUERY data, decision making, checking for suspects in in_generated_paths_list, etc. The complete process is explained in steps in this section.

For every suspect in the suspects_list, apply the following steps:

Step 1: Get the suspect's FORWARD NODE value. The FORWARD NODE value is significant because the SN wants to know to whom the suspect node will forward the

packets to if the SN were to send data packets to it. The FORWARD NODE value can be found with the RREP control packet that the SN receives from the suspect. The SN then buffers the FORWARD NODE value along with the PATH value in the `in_generated_paths_list`.

Step 2: Generate an RQUERY message and send it to the node found in FORWARD NODE variable. Set the parameters for RQUERY as follows: Packet Id is a randomly generated Id that uniquely identifies this RQUERY message; Path is the absolute path from SN to the FORWARD NODE and can be found in either the `in_generated_paths_list` or `dn_generated_paths_list`. The RQUERY control message has to follow this path in order to get to the FORWARD NODE; Suspect Node is the current node in question from the `suspects_list`. Destination is SN's destination to which it intends to communicate with; `Is_Suspect_Neighbour` is a Boolean TRUE or FALSE. This value is filled by the FORWARD NODE once the RQUERY is received by it; `Is_Path_to_Destination` is also a Boolean TRUE or FALSE and is filled by the FORWARD NODE once the RQUERY is received by it.

Step 3: Upon receiving the RQUERY message back from the FORWARD NODE, get the `Is_Suspect_Neighbour` and `Is_Path_to_Destination` values.

Step 4: If `Is_Suspect_Neighbour` is FALSE, FORWARD NODE is not a neighbour of suspect node, so there is no path from suspect to the FORWARD NODE. We can assume that the FORWARD NODE value that the suspect has sent in RREP is invalid, and therefore, we conclude that the suspect is a BLACK HOLE. If the `Is_Suspect_Neighbour` is TRUE, go to step 5.

Step 5: If *Is_Path_to_Destination* is FALSE, then when SN sends data packets to suspect, it will forward the data to a node which does not have a route to the destination. From this, we can assume that the information that suspect provided is misleading, and we can conclude that the suspect is a BLACK HOLE. If *Is_Path_to_Destination* is TRUE, go to step 6.

Step 6: If *Is_Path_to_Destination* is TRUE, we will check if there is a path in *in_generated_paths_list* which goes through the suspect. If there is such a path, then we conclude that suspect is NOT a BLACK HOLE because a black hole never forwards any packets. Therefore, it should not have a path going through it. Otherwise, we conclude that the suspect is a BLACK HOLE.

Figure 8 and Figure 9 represents the pseudo code and block diagram of the FRBD algorithm respectively. Both these figures are shown below:

```
1  Get suspects_list
2  FOREACH suspect in suspects_list
3    Get suspect's FORWARD NODE value from in_generated_paths_list
4    Set query_message = Send RQUERY to FORWARD NODE and Get response
5    Get Is_Suspect_Neighbour and Is_Path_to_Destination from query_message
6    IF Is_Suspect_Neighbour FALSE
7      print, suspect is a BLACKHOLE
8    ELSE IF Is_Suspect_Neighbour TRUE
9      IF Is_Path_to_Destination FALSE
10     print, suspect is a BLACKHOLE
11     ELSE IF Is_Path_to_Destination TRUE
12       Set result = check for Path going through suspect in
in_generated_paths_list
13       IF result is FALSE
14         print, suspect is a BLACKHOLE
15       ELSE IF result is TRUE
16         print, suspect is NOT a BLACKHOLE
17       END IF
18     END IF
19   END IF
20 END FOR
```

Figure 8. Pseudo Code – Forward Reaching Black Hole Detection Algorithm

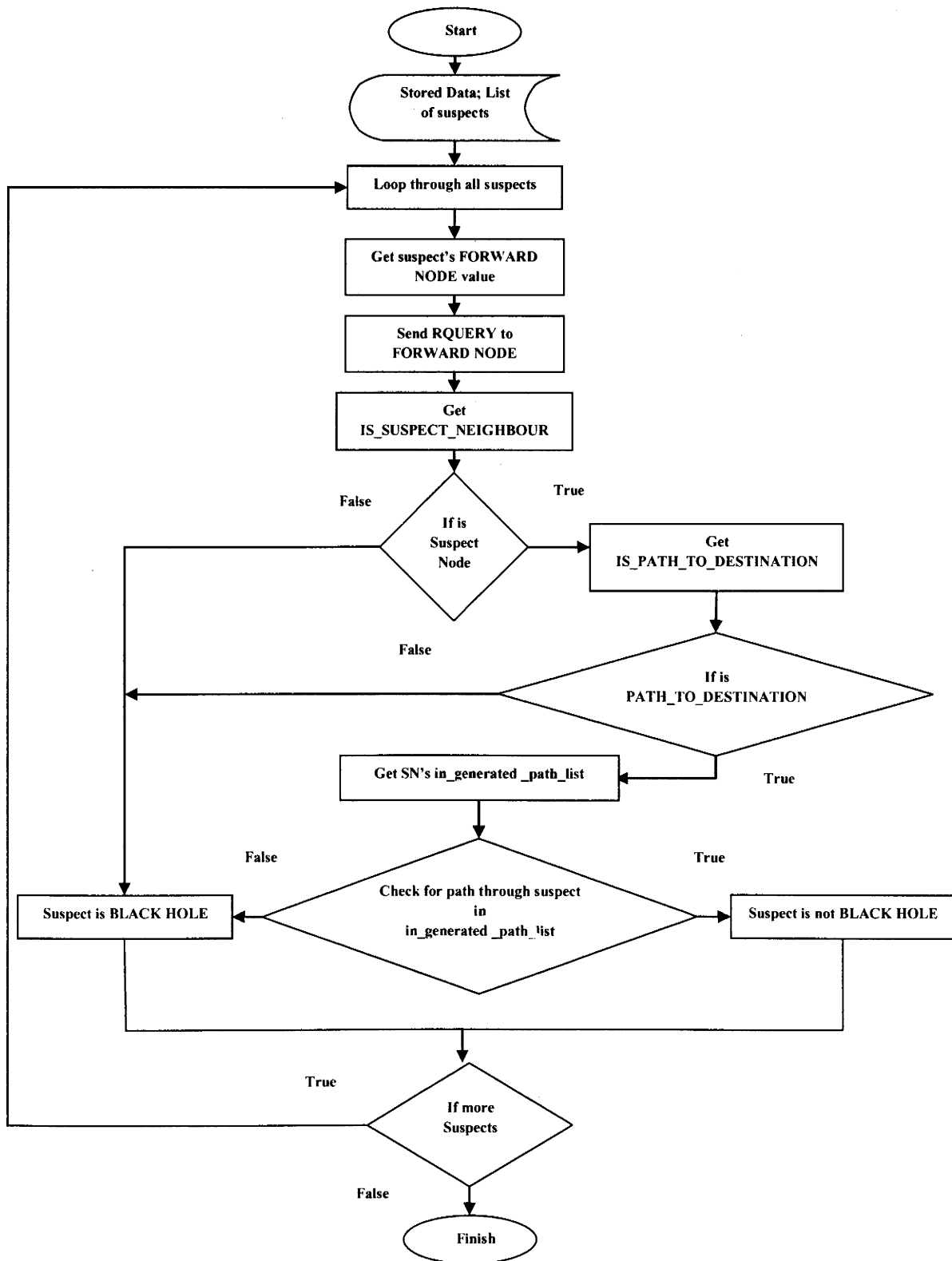


Figure 9. Forward Reaching Black Hole Detection Algorithm Block Diagram

CHAPTER 4. RESULTS

The experimental setup requires a set of 10 different network layouts. The FRBD algorithm is applied over these network layouts to test their effectiveness in successfully detecting a black hole. A Java based simulation tool has been specifically developed for this purpose (Refer Chapter 5: Simulation Tool). Each of these network layouts consist of wireless nodes and paths connecting these nodes. They differ in parameters such as path density and the type of black hole node – Multi Connected, Isolated Leaf and Isolated.

The path density of the network layout can be altered by varying the number of nodes while keeping the sensor range constant OR by changing the sensing range and keeping the number of nodes constant.

The Multi Connected Node is a type of black hole node which has more than one neighbor directly connected to it. In Figure 10, Node 22 is a Multi Connected Black Hole Node.

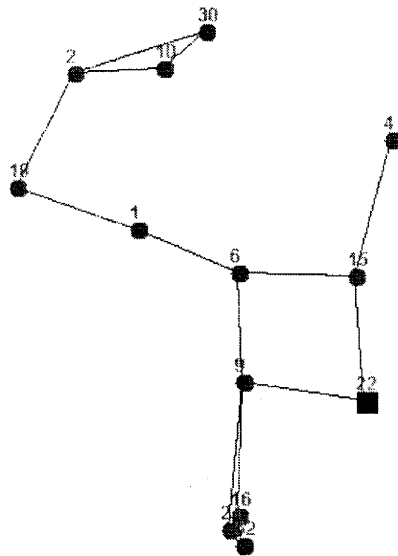


Figure 10. Multi Connected Black Hole Node

The Isolated Leaf Node is a type of black hole node which has just one neighbor directly connected to it. In Figure 11, Node 31 is an Isolated Leaf Black Hole Node.

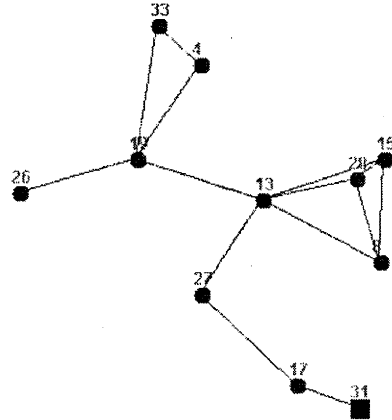


Figure 11. Isolated Leaf Black Hole Node

The Isolated Node is a type of black hole node which does not receive the AODV control packets due to its remote positioning in the network layout. In Figure 12, Node 32 (Isolated Node) is seen as being completely detached from the other nodes.

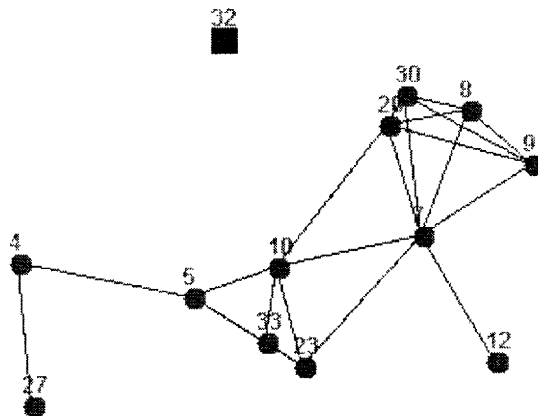


Figure 12. Isolated Black Hole Node

4.1. Test Cases

The input and result parameters are listed below:

N_N = Total Number of Nodes in the Network Layout.

S_R = Sensor Range between Nodes.

N_P = Total Number of Paths in the Network Layout.

S_{ID} = Source Node, ID of the node that initiates the route discovery process.

D_{ID} = Destination Node, ID of the node to which S intends to communicate with.

B_H = Black Hole Node, ID of the malicious node that eats up the packets intended for D.

B_H Type = Type of Black Hole Node (Refer Figures 10, 11 and 12).

S_{ID} = Suspect Node ID.

D_S Time = Discovery Time (in nanoseconds), Time taken by FRBD algorithm to verify the suspect node.

For experimental purposes, we have conducted ten test runs. The inputs and results for each of these test runs are shown below in Tables 1 through 10. These test cases have been randomly generated by the simulation tool. They vary in the density of paths (N_P / N_N). Our attempt is to test the effectiveness of FRBD algorithm in sparse as well as dense path environments. The test run tables also contain the time taken by the FRBD algorithm to process the suspect node.

In test case 1, shown in Table 1 below, the number of nodes (N_N) is 25 and the total number of paths in the network (N_P) is 9. The source node (S_{ID}) is 15, destination node (D_{ID}) is 18 and the black hole (B_H) node is 23. For this test case, a total of 2 suspects were generated and the results are presented in Table 1.

Table 1. Test Case # 1 (Multi Connected Black Hole, 2 Suspects, Black Hole Found)

N_N	S_R	N_P	S_{ID}	D_{ID}	B_H	B_H Type	S_{ID}	D_S Time	Black Hole Found
25	100.0	9	15	18	23	Multi Connected	6	435531	No
							23	277410	Yes

In test case 2, shown in Table 2 below, the number of nodes (N_N) is 35 and the total number of paths in the network (N_P) is 24. The source node (S_{ID}) is 6, destination node (D_{ID}) is 4 and the black hole (B_H) node is 11. For this test case, a total of 3 suspects were generated and the results are presented in Table 2.

Table 2. Test Case # 2 (Multi Connected Black Hole, 3 Suspects, Black Hole Found)

N_N	S_R	N_P	S_{ID}	D_{ID}	B_H	B_H Type	S_{ID}	D_S Time	Black Hole Found
35	100.0	24	6	4	11	Multi Connected	9	730540	No
							21	252546	No
							11	1170260	Yes

In test case 3, shown in Table 3 below, the number of nodes (N_N) is 25 and the total number of paths in the network (N_P) is 46. The source node (S_{ID}) is 10, destination node (D_{ID}) is 12 and the black hole (B_H) node is 21. For this test case, a total of 4 suspects were generated and the results are presented in Table 3.

Table 3. Test Case # 3 (Isolated Leaf Black Hole, 4 Suspects, Black Hole Found)

N_N	S_R	N_P	S_{ID}	D_{ID}	B_H	B_H Type	S_{ID}	D_S Time	Black Hole Found
25	150.0	46	10	12	21	Isolated Leaf	17	844521	No
							7	379378	No
							21	906261	Yes
							18	232432	No

In test case 4, shown in Table 4 below, the number of nodes (N_N) is 65 and the total number of paths in the network (N_P) is 144. The source node (S_{ID}) is 22, destination node (D_{ID}) is 60 and the black hole (B_H) node is 41. For this test case, a total of 19 suspects were generated and the results are presented in Table 4.

Table 4. Test Case # 4 (Isolated Leaf Black Hole, 19 Suspects, Black Hole Found)

N_N	S_R	N_P	S_{ID}	D_{ID}	B_H	B_H Type	S_{ID}	D_S Time	Black Hole Found
65	100.0	144	22	60	41	Isolated Leaf	44	961575	No
							13	907936	No
							37	92190	No
							62	9805995	No
							16	360102	No
							10	1598806	No
							24	1380064	No
							0	36317	No
							59	445308	No
							33	159517	No
							49	4469004	No
							61	11454	No
							35	12951367	No
							19	20015114	No
							27	13400027	No
							25	8814249	No
							50	131861	No
							41	18584765	Yes
							55	7115709	No

In test case 5, shown in Table 5 below, the number of nodes (N_N) is 35 and the total number of paths in the network (N_P) is 89. The source node (S_{ID}) is 5, destination node (D_{ID}) is 10 and the black hole (B_H) node is 2. For this test case, a total of 13 suspects were generated and the results are presented in Table 5.

Table 5. Test Case # 5 (Multi Connected Black Hole, 13 Suspects, Black Hole Found)

N_N	S_R	N_P	S_{ID}	D_{ID}	B_H	B_H Type	S_{ID}	D_S Time	Black Hole Found
35	150.0	89	5	10	2	Multi Connected	18	411784	No
							2	599238	Yes
							19	11636116	No
							9	839772	No
							12	2791416	No
							4	211480	No
							34	3924521	No
							25	220419	No
							13	11917716	No
							6	429943	No
							7	4500292	No
							32	521854	No
							14	4427658	No

In test case 6, shown in Table 6 below, the number of nodes (N_N) is 45 and the total number of paths in the network (N_P) is 67. The source node (S_{ID}) is 5, destination node (D_{ID}) is 29 and the black hole (B_H) node is 3. For this test case, a total of 6 suspects were generated and the results are presented in Table 6.

Table 6. Test Case # 6 (Isolated Leaf Black Hole, 6 Suspects, Black Hole Found)

N_N	S_R	N_P	S_{ID}	D_{ID}	B_H	B_H Type	S_{ID}	D_S Time	Black Hole Found
45	100.0	67	5	29	3	Isolated Leaf	24	635276	No
							21	417092	No
							42	297524	No
							17	1792686	No
							19	440280	No
							3	221537	Yes

In test case 7, shown in Table 7 below, the number of nodes (N_N) is 55 and the total number of paths in the network (N_P) is 104. The source node (S_{ID}) is 47, destination node

(D_{ID}) is 31 and the black hole (B_H) node is 17. For this test case, a total of 17 suspects were generated and the results are presented in Table 7.

Table 7. Test Case # 7 (Isolated Black Hole, 17 Suspects, Black Hole Not Found)

N _N	S _R	N _P	S _{ID}	D _{ID}	B _H	B _H Type	S _{ID}	D _S Time	Black Hole Found
55	100.0	104	47	31	17	Isolated	18	60343	No
							43	700368	No
							6	1192889	No
							26	232432	No
							21	34641	No
							7	218184	No
							12	600914	No
							32	29333	No
							44	2432432	No
							38	1608863	No
							14	8929067	No
							42	3013232	No
							2	5128305	No
							52	8092648	No
							35	7207899	No
							39	9119036	No
							25	6365334	No

In test case 8, shown in Table 8 below, the number of nodes (N_N) is 65 and the total number of paths in the network (N_P) is 441. The source node (S_{ID}) is 54, destination node (D_{ID}) is 1 and the black hole (B_H) node is 18. For this test case, a total of 0 suspects were generated and the results are presented in Table 8.

Table 8. Test Case # 8 (Multi Connected Black Hole, 0 Suspect, Black Hole Not Found)

N _N	S _R	N _P	S _{ID}	D _{ID}	B _H	B _H Type	S _{ID}	D _S Time	Black Hole Found
65	150.0	441	54	1	18	Multi Connected	N/A *	N/A *	N/A *

In test case 9, shown in Table 9 below, the number of nodes (N_N) is 55 and the total number of paths in the network (N_P) is 278. The source node (S_{ID}) is 25, destination node (D_{ID}) is 52 and the black hole (B_H) node is 40. For this test case, a total of 24 suspects were generated and the results are presented in Table 9.

Table 9. Test Case # 9 (Isolated Leaf Black Hole, 24 Suspects, Black Hole Found)

N_N	S_R	N_P	S_{ID}	D_{ID}	B_H	B_H Type	S_{ID}	D_S Time	Black Hole Found
55	150.0	278	25	52	40	Isolated Leaf	43	767696	No
							23	422400	No
							33	738920	No
							13	23706924	No
							7	270705	No
							31	3584534	No
							28	3684547	No
							30	4070071	No
							41	3114642	No
							19	15478504	No
							12	562083	No
							27	15753120	No
							46	946210	No
							16	15410897	No
							4	935873	No
							15	22694225	No
							47	23556346	No
							26	15096053	No
							44	23435940	No
							32	4933029	No
							35	16598479	No
							51	14445970	No
							8	22786975	No
							40	16589818	Yes

In test case 10, shown in Table 10 below, the number of nodes (N_N) is 45 and the total number of paths in the network (N_P) is 160. The source node (S_{ID}) is 7, destination node

(D_{ID}) is 30 and the black hole (B_H) node is 26. For this test case, a total of 15 suspects were generated and the results are presented in Table 10.

Table 10. Test Case # 10 (Isolated Black Hole, 15 Suspects, Black Hole Not Found)

N_N	S_R	N_P	S_{ID}	D_{ID}	B_H	B_H Type	S_{ID}	D_S Time	Black Hole Found
45	150.0	160	7	30	26	Isolated	17	1180876	No
							34	756520	No
							10	1123327	No
							19	353117	No
							22	222375	No
							40	443353	No
							18	3938769	No
							11	13689	No
							39	3514693	No
							1	290261	No
							31	2934172	No
							5	877486	No
							41	671594	No
							28	5671671	No
							44	4567899	No

* Note :- AMPL student addition is limited to 300 variables and 300 constraint the above problem has 352 variables and 63 constraints.

4.2. Analysis of Results

The FRBD algorithm accepts a set of suspects as input and processes each one individually to find whether the suspect in question is a black hole. The results table consists of three fields namely S_{ID} , D_S Time and Black Hole Found. The S_{ID} is the suspect id that was passed into the FRBD algorithm. D_S Time is the time taken by the FRBD algorithm to process S_{ID} . The Black Hole Found field consists of Yes/ No values depending on whether the S_{ID} is a black hole or not.

As an example, we will study one of the results of test case # 9 shown in Table 9 above. In Table 9, when input parameters were passed to the Modified AODV protocol, a set of 24 suspects were generated. These suspects were then passed to the FRBD algorithm. As we can see from Table 9, it took FRBD algorithm 4070071 ns to process node 30. The result was that node 30 is not a black hole. The other suspects are also interpreted the same way as node 30.

The test cases above have shown that whenever a black hole was of type Multi Connected or Isolated Leaf, it was successfully detected by the FRBD algorithm. The only case where the FRBD algorithm couldn't detect the black hole is when the black hole itself was of type Isolated. Not being able to detect Isolated black holes is not a concern as such nodes are completely detached from the main network and so never receive any data packets. So the threat perception from these kinds of black holes is negligible. Out of the total ten test cases executed, eight were of type Isolated Leaf or Multi Connected black hole nodes and out of these eight test cases, the FRBD algorithm could successfully detect a black hole in all these eight cases. From the results of these test cases, we can assume that the accuracy for FRBD algorithm is 100 %.

CHAPTER 5. SIMULATION TOOL

A tool was designed and developed to simulate the performance of various types of test problems. These test problems differ in the total number of nodes as well as the sensing range of each node. The results for all these different problems are analyzed to evaluate the performance of the described approach.

5.1. Development Environment

- Java version: 1.6.0_11
- Integrated Development Environment: Netbeans 6.7
- Operating System: Windows XP Profession x86 – Service Pack 3
- AMPL: Windows standard Student Edition
- AMPL Solver: MINOS 5.5

5.2. Tool

5.2.1. Inputs

The simulation tool accepts two types of inputs for generating a network layout.

- External (.csv format) file.

The network layout can be generated by importing a csv format file containing the coordinate information for each nodes.

- Randomly generated set of nodes.

The network layout can also be generated by specifying a count for the number of nodes. These nodes are then randomly generated by the program. The program allows the user to save a randomly generated network to the system (in .csv format) and reload it at a later point in time.

Once the numbers of nodes are specified, the simulation tool randomly generated these nodes and this layout acts as an input for the Forward Reaching Black Hole Detection (FRBD) algorithm.

5.2.2. Outputs

The program outputs the following information:

- A list of paths generated by the modified AODV protocol in response to a source initiated path discovery process.
- A list of paths sent as an input to the AMPL set coverage algorithm.
- A list of paths generated as output by the AMPL set coverage algorithm.
- A set of suspect nodes on which the source initiates the Forward Reaching Black Hole Detection (FRBD) algorithm.
- Result of the FRBD algorithm (Black Hole Found? Yes/ No).

The program also outputs the time taken to find the Black Hole (CPU time in nanoseconds) which is required for evaluating the performance of the FRBD algorithm.

5.2.3. Software Domain Design

The software domain design in this section consists of a basic use case diagram, class diagram overview, class diagram detailed and a sequence diagram. The design aspect is important to understand the internals as well as the interaction between various components. The advantage of software architecture is that it provides basis for capture and reuse of design knowledge. The software domain design enables to represent a complex system in easy to use manner.

5.2.3.1. Basic Use Case Diagram

The use case diagram in Figure 13 provides a basic overview of the system.

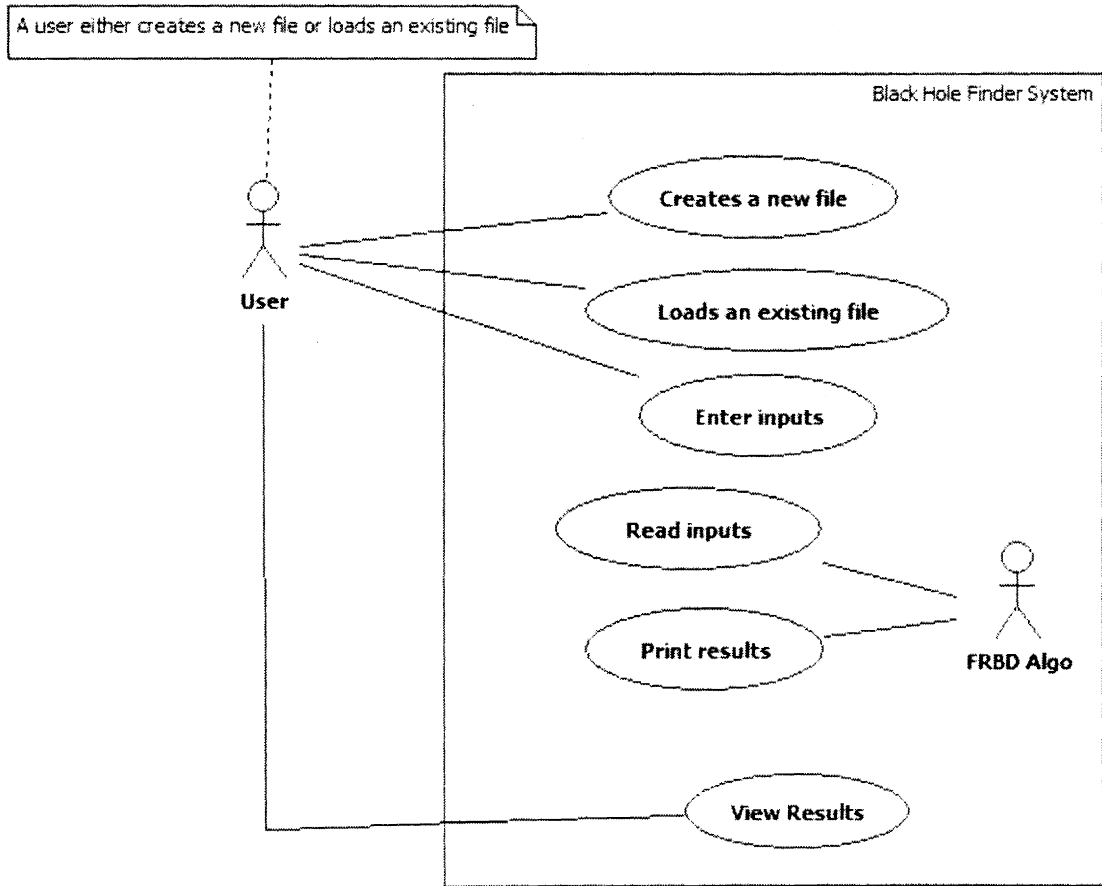


Figure 13. Basic Use Case Diagram

5.2.3.2. Class Diagram Overview

Figure 14 shows an overview of the classes, including packages in the program.

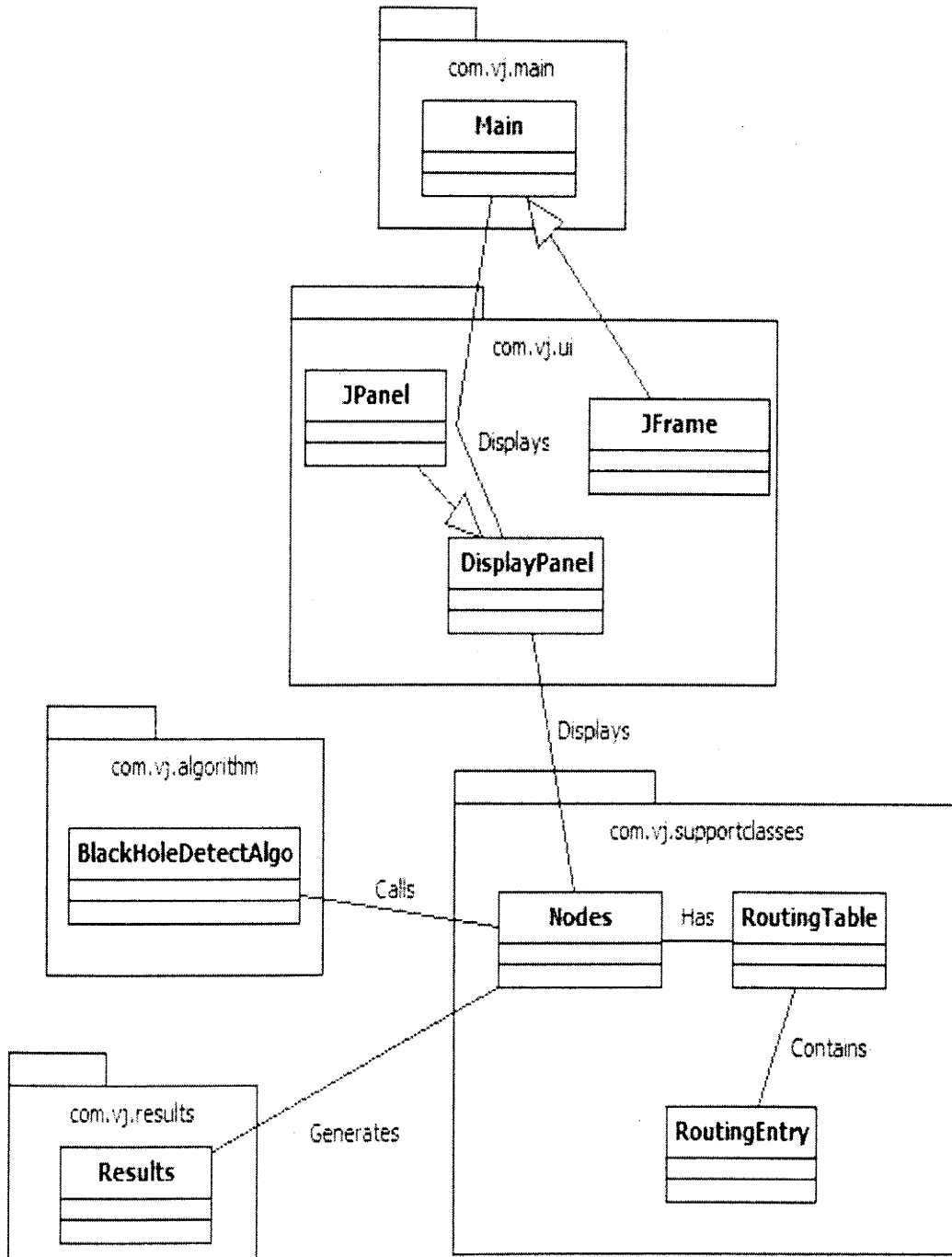


Figure 14. Class Diagram Overview

5.2.3.3. Class Diagram Detailed

The class diagram in Figure 15 shows details about each attributes and operations in the classes.

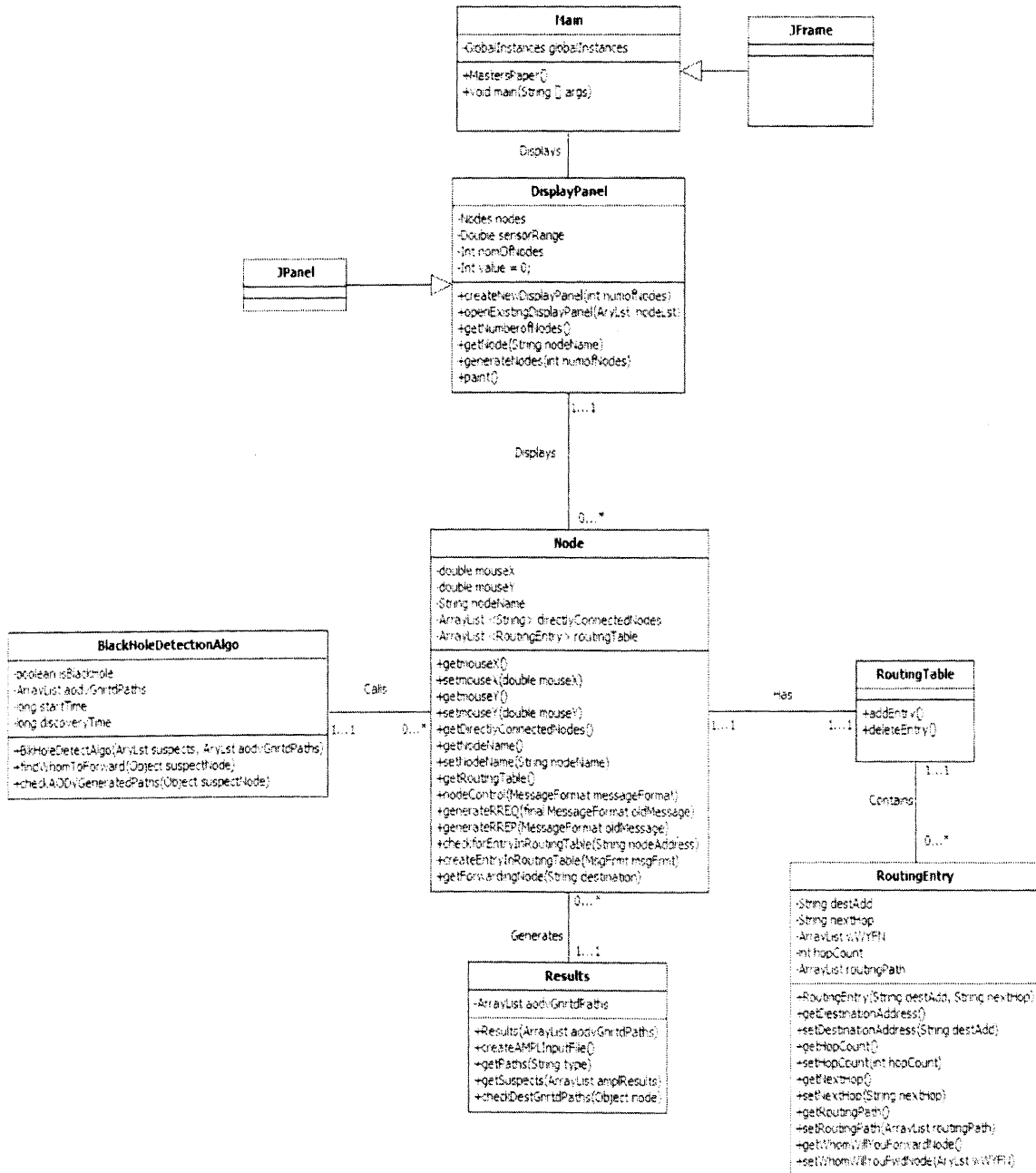


Figure 15. Class Diagram Detailed

5.2.3.4. Sequence Diagram

The Sequence Diagram in Figure 16 represents the interaction of the user with the Black Hole Finder System.

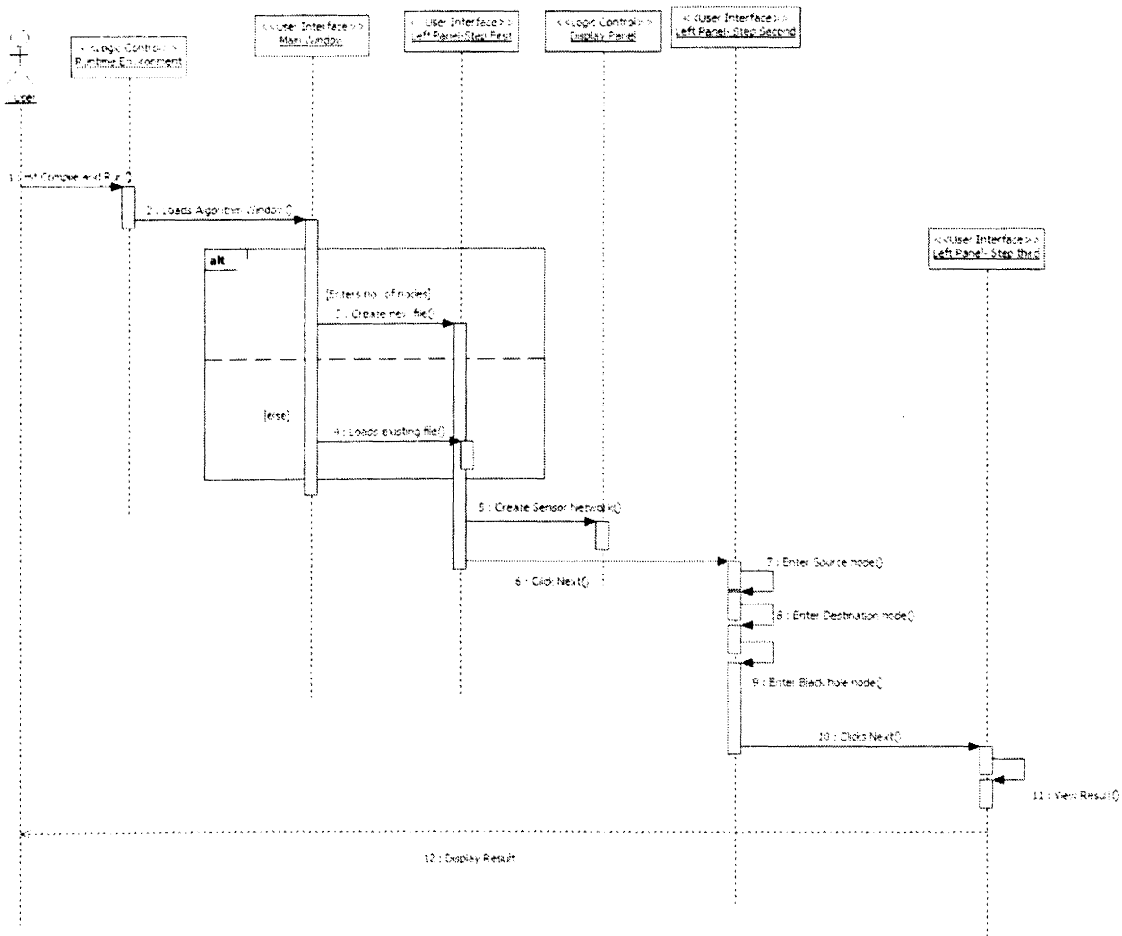


Figure 16. Sequence Diagram

5.2.4. User Interface Design

5.2.4.1. Overview

The simulation tool has a very simple and user friendly interface design. The process of loading a network layout file through running the FRBD algorithm takes place in three easy steps. In the first step, the user is prompted to load the data required to generate the

network layout. In the second step, the user enters the source, destination and black hole address in the provided text fields and clicks next. In the third step, the user clicks the view results button and the results appear in the text area provided for the purpose of displaying results. Figure 17 displays the user interface of the simulation tool. The left panel of the UI shows the three step process, the right panel is the main display panel of the tool, and the green dots represent a node and the arcs between nodes are the connectivity.

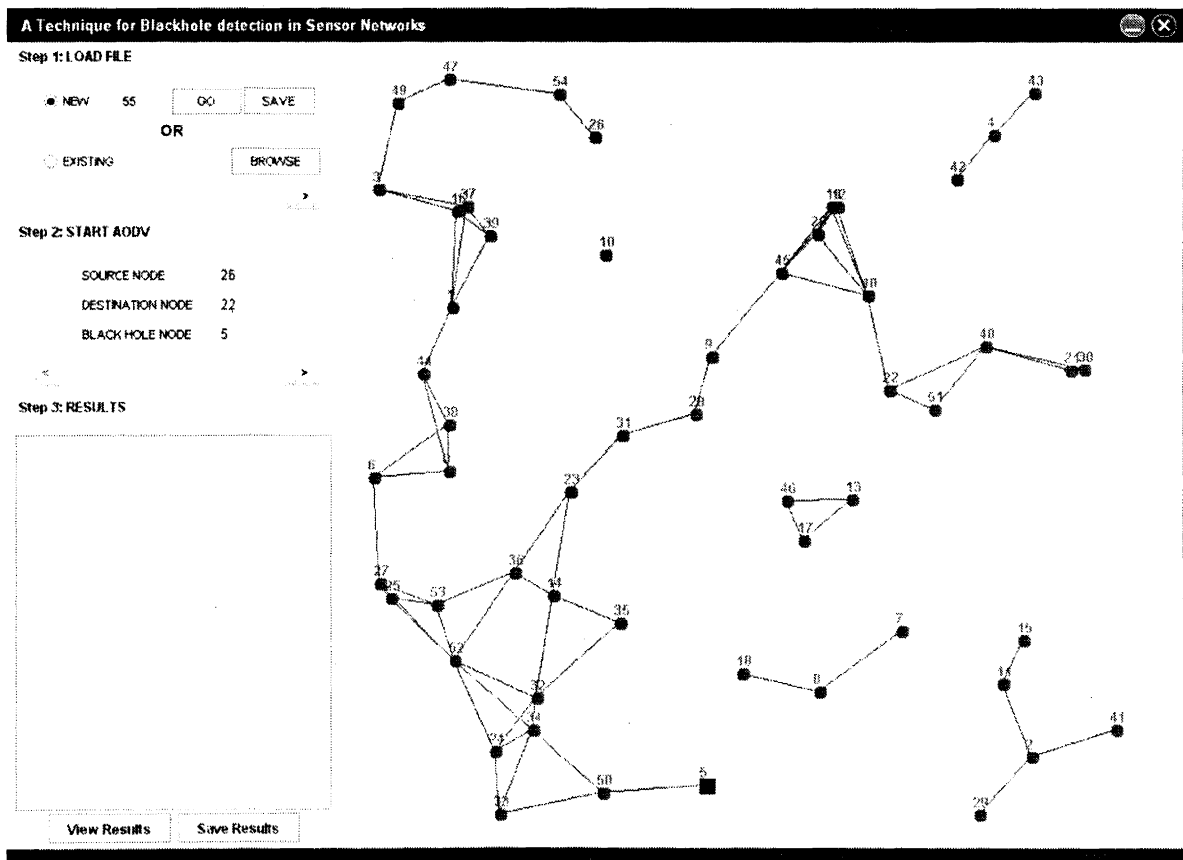


Figure 17. Simulation Tool User Interface

The red colored square node represents a black hole. The following section examines the navigational steps in detail.

5.2.4.2. Navigational Steps

Step 1: Load File

The user has two options for loading nodes data in to the program, both of which are briefly described below.

- **From External File:** The user has an ability to browse the data file from the local system using the file chooser and to select the required .csv format file. To load an external file, it has to be of .csv extension and it should contain attributes in the order mouse X value, mouse Y value and the node name. For every node in the node layout network, these three attributes must be present in the .csv file. Figure 18 shows a sample input file used to generate the nodes network.

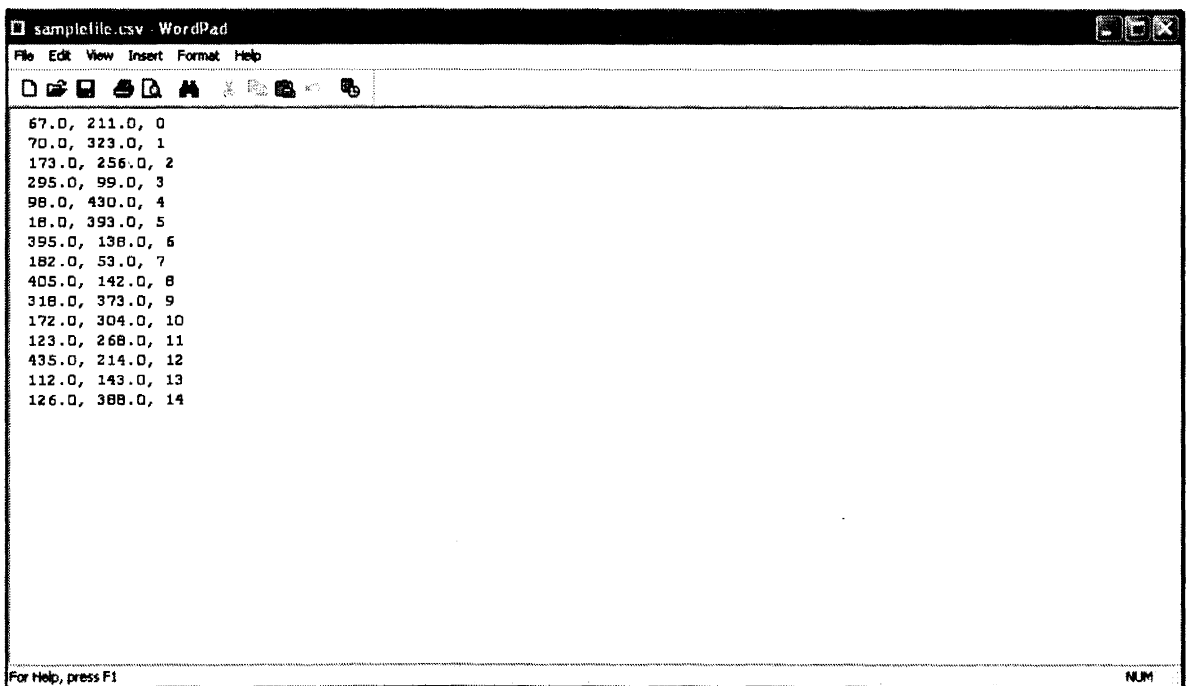


Figure 18. Simulation Tool Input File to Generate Nodes Network

- **Randomly Generated Nodes:** The program has the required functionality to generate a network layout. The user needs to specify the number of nodes. These nodes are then generated with the help of a random function present in the program.

Once the node coordinates are generated, the program paints these coordinates on the designated location in the user interface.

Step 2: Set Parameters

In the second step, the user is required to enter the source, destination and black hole node address in the provided text fields. These parameters are the inputs to the modified AODV protocol. Based on these values, the modified AODV protocol generates a set of paths between the source and the destination nodes.

Step 3: View Results

When the user clicks next in Step 2, the set parameters are sent to the AODV protocol and the paths are generated. These paths are then sent to the AMPL program which implements the set coverage problem on these paths. The AMPL program returns a subset of paths that were provided to it as an input, and these paths are then sent to the FRBD algorithm which tries to find the black hole among these paths. All the mentioned process takes place in the background. In this step, the user is provided with a “View Results” button. On clicking this button, all the inputs, outputs as well as performance parameters are printed on to the results text area. Figure 19 depicts a sample results output file.

5.3. AMPL Set Coverage Algorithm

AMPL is an algebraic modeling language for linear and nonlinear optimization problems, in discrete or continuous variables. This section is an extension of the set coverage problem described in section 4.4.2.1, presents an example of a set coverage problem, and then describes the steps involved in executing that problem. It also provides an overview of the input / output file formats, AMPL commands and interfacing AMPL with JAVA.

```

results.txt WordPad
File Edit View Insert Format Help
[Icons]
Paths in SN's dn_generated_paths_list
path 1: [14, 5, 4]
path 2: [14, 7, 4]
path 3: [14, 12, 5, 4]

Paths in SN's in_generated_paths_list
path 1: [9, 5, 4]
path 2: [9, 4]
path 3: [9, 7, 4]
path 4: [7, 5, 4]
path 5: [5, 7, 4]
path 6: [7, 12, 5, 4]
path 7: [9, 12, 5, 4]
path 8: [12, 7, 4]

Initial Suspects: [9, 7, 5, 12]

Ampl result paths:
path 1: [9, 7, 4]
path 2: [7, 12, 5, 4]
path 3: [9, 12, 5, 4]

suspects: [9]
suspect: 9 start time: 271186760793341
suspect: 9 discovery time: 212876
BlackHole found! Node 9 is a blackhole
For Help, press F1

```

Figure 19. Results Output File

5.3.1. Sample Problem

Assume that there are 16 nodes in the network (0, 1, 2 ...15) and the following are the paths in the SN's (node 4) *in_generated_paths_list* buffer (Figure 20).

```

path 1: [9, 5, 4]
path 2: [9, 4]
path 3: [9, 7, 4]
path 4: [7, 5, 4]
path 5: [5, 7, 4]
path 6: [7, 12, 5, 4]
path 7: [9, 12, 5, 4]
path 8: [12, 7, 4]

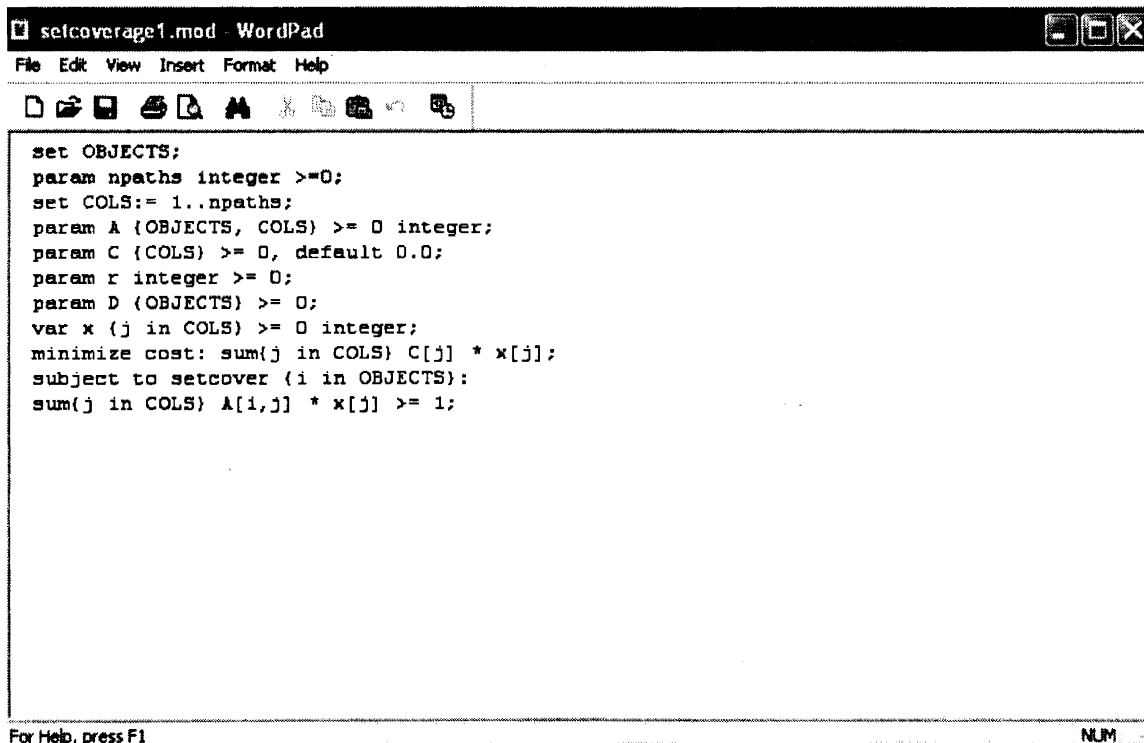
```

Figure 20. Sample Paths in Source's *in_generated_paths_list* buffer

The *rrep_generator_node* are the suspects, and with 8 paths in *in_generated_paths_list*, the suspects are {9, 7, 5, 12}.

5.3.1.1. Input File

The AMPL required two kinds of input files (model and data) to execute the process. The model file consists of the linear program formulation, where as the data file consists of the input required to execute the linear program formulation.



```
set OBJECTS;
param npaths integer >=0;
set COLS:= 1..npaths;
param A {OBJECTS, COLS} >= 0 integer;
param C {COLS} >= 0, default 0.0;
param r integer >= 0;
param D {OBJECTS} >= 0;
var x {j in COLS} >= 0 integer;
minimize cost: sum{j in COLS} C[j] * x[j];
subject to setcover {i in OBJECTS}:
sum{j in COLS} A[i,j] * x[j] >= 1;
```

Figure 21. AMPL Model File

In the model file, the OBJECTS are the number of nodes, and npaths are the number of paths sent to the AMPL algorithm (refer to Figure 21). In this example, the number of nodes is 16 and the number of paths is 8. C is the cost parameter, and every path is associated with the cost parameter. We consider the number of hops as the cost parameter, so for path 1, the cost parameter is 3, for path 2, the cost parameter is 2, and so on. Figure 22 shows a sample data file. Apart from the OBJECTS, npaths and C, it also

consists of a table (parameter A) with values {0, 1}. Each row in the table denotes a path, and each column denotes the node name.

```

data:
set OBJECTS:= 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
param npaths:= 8;
param C [*] :=
1 3
2 2
3 3
4 3
5 3
6 4
7 4
8 3
;

param A[*,*] (cr) default 0:
      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 :=
1 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0
2 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
3 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0
4 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0
5 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 0 0 0 0
7 0 0 0 0 0 1 1 0 0 0 1 0 0 1 0 0 1 0 0
8 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0
;

```

Figure 22. AMPL Data File

For every path (row), if there is a node (column) that belongs to the path, we add a value 1 in that intersection, otherwise the value is 0. For example, consider path 1: {9, 5, 4}. We will add a 1 in the intersection of (1, 9), (1, 5) and (1, 4) respectively. All the other intersections for path 1 are set to 0 (refer to Figure 22). The same process is applied to the complete table.

5.3.1.2. The Process

Both data file and model file are loaded in to the AMPL command line. The free AMPL student version is available for download in the AMPL website at <http://www.ampl.com>. There are various solvers that are available with AMPL. In this example, we will use MINOS which is the default solver of AMPL. It is required that the names of data and

model file have to be same although their extensions will be different (.mod and .dat). The following are the commands used to load a file, send data to solver, retrieve data from solver, and display results (refer to Figure 23).

A screenshot of a Windows command prompt window titled "C:\amplcm1\ampl.exe". The window contains the following text:

```
ampl: include modinc;  
ampl: model setcoverage1.mod;  
ampl: data setcoverage1.dat;  
ampl:
```

Figure 23. Screenshot AMPL Command Line – Loading Model and Data Files

- include modinc: AMPL include command for easy access to the MODELS and MODELS/NLMODELS subdirectories (Refer to Figure 23).
- model setcoverage1.mod: AMPLmodel command is for loading a model file (Refer to Figure 23).
- data setcoverage1.dat: AMPL data command is for loading a data file (Refer to Figure 23).
- solve: AMPL solve command sends the linear program to a linear programming solve (MINOS.exe) and returns the answer (Refer to Figure 24).

A screenshot of a Windows command prompt window titled "C:\amplcm1\ampl.exe". The text inside the window shows the execution of the 'solve' command. The output from the MINOS 5.5 solver is displayed, indicating that it ignored integrality for 8 variables and found an optimal solution with an objective value of 5.5 after 2 iterations. The prompt returns to 'ampl:'.

```
C:\amplcm1\ampl.exe
ampl: solve;
MINOS 5.5: ignoring integrality of 8 variables
MINOS 5.5: optimal solution found.
2 iterations, objective 5.5
ampl:
```

Figure 24. Screenshot AMPL Command Line-Solve Command

- display: AMPL display command is used to show the optimal values of the variables (refer to Figure 25).

A screenshot of a Windows command prompt window titled "C:\amplcm1\ampl.exe". The text shows the 'display' command being used to show the optimal values for variables x[1] through x[8]. The output lists the values: x[1]=0, x[2]=0, x[3]=0.5, x[4]=0, x[5]=0, x[6]=0.5, x[7]=0.5, and x[8]=0. The prompt returns to 'ampl: _'.

```
C:\amplcm1\ampl.exe
ampl: solve;
MINOS 5.5: ignoring integrality of 8 variables
MINOS 5.5: optimal solution found.
2 iterations, objective 5.5
ampl: display x;
x [*] :=
1 0
2 0
3 0.5
4 0
5 0
6 0.5
7 0.5
8 0
;
ampl: _
```

Figure 25. Screenshot AMPL Command Line – Display Command

In Figure 25, the value of x is displayed using the AMPL display command. Each row represents a path. The value for each row is either a zero or non zero element, and all the nonzero values are considered to be the AMPL result. For a given set of 8 paths, the results for AMPL set coverage algorithm are paths 3, 6 and 7. Comparing to the corresponding input paths (refer to Figure 20), the AMPL output paths (results) are –

path 3: {9, 7, 4}
path 6: {7, 12, 5, 4}
path 7: {9, 12, 5, 4}

Figure 26. AMPL Set Coverage Results

After applying the AMPL set coverage to the initial problem, the new suspects are {9, 7}. Compared to the original problem, the suspects list has been optimized to just two nodes; as a result, the FRBD algorithm is more efficient in finding black holes.

5.3.2. Interfacing AMPL with JAVA

AMPL provides a possibility of accessing its command line through the JAVA GUI. The access code is distributed as JAVA source. The AMPL and the appropriate solver have to be downloaded separately. The advantage of accessing AMPL through JAVA is that it reduces the burden of loading the files manually from the Java program to the AMPL command line and vice versa. The complete process of sending and receiving data from the JAVA program to AMPL is automated.

The JAVA program creates a process object from runtime to access the AMPL. The AMPL executable (ampl.exe) is then accessed from the local machine, and the data between JAVA program and AMPL is sent in the form of bytes. The process objects thus opens an input / output stream for the byte transmission. The sample interfacing code is provided below-

```
String[] ampcmd = new String[2];           // parameters array
ampcmd[0] = "C:\\ampcmd\\amp.exe";         // path to amp.exe in local machine
ampcmd[1] = "-b";                          // turns on block mode, off cr
Runtime rt = Runtime.getRuntime();        // getting runtime object
Process p = rt.exec(ampcmd);               // creating a process with AMPL
p.getInputStream();                         // receiving bytes from AMPL
p.getOutputStream();                       // sending bytes to AMPL
```

Figure 27. Setting up Connection Between AMPL and JAVA

CHAPTER 6. CONCLUSIONS AND FUTURE WORK

This paper discusses the security issues involved while using the AODV protocol on MANETs, describes the single black hole attacks, and proposes a solution to identify these black holes in MANETs. The proposed solution can be applied to finding a single black hole in the network. Once the black hole has been identified, the source discovers a secure path to the destination by avoiding a black hole.

As a part of my future work, I plan to extend the Forward Reaching Black Hole Detection Algorithm to identify multiple black hole nodes acting in cooperation in MANETs.

REFERENCES

- ADSL Gate. (n.d.). <http://www.adslgate.com/>. Retrieved October 2, 2010, from <http://www.adslgate.com/dsl/showthread.php?t=586380>
- Al-Shurman, M., Yoo, S.-M., & Park, S. (2004). Black Hole Attack in Ad Hoc Networks. *ACM Southeast Regional Conference* (pp. 96-97). Huntsville: ACM.
- Bounpadith, K., Nakayama, H., Nemoto, Y., & Kato, N. (2007). A survey of routing attacks in mobile ad hoc networks. *IEEE Wireless Communications Magazine*, 14, pp. 85-91. IEEE Communications Society.
- D-Link. (n.d.). <http://www.dlink.com.au/>. Retrieved October 2, 2010, from http://www.dlink.com.au/tech/faq/wireless/wireless_faq_general.htm
- Fourer, R., & Gay, D. (1995). Expressing Special Structures in an Algebraic Modeling Language for Mathematical Programming. *ORSA Journal on Computing*, 7 (ISSN 0899-1499), 166-190.
- Mistry, N., Jinwala, D., & Zaveri, M. (2010). Improving AODV Protocol against Blackhole Attacks. *International MultiConference of Engineers and Computer Scientists* (pp. 1040-1045). Hong Kong: IAENG International Journal of Computer Science.
- Peng, N., & Kun, S. (2003). How to Misuse AODV: A Case Study of Insider Attacks against Mobile Ad-hoc Routing Protocols. *Technical Report: TR-2003-07*, 3, pp. 60-67. Raleigh: North Carolina State University.
- Perkins, C., & Royer, E. (1999). Ad-Hoc On-Demand Distance Vector Routing. *Mobile Computing Systems and Applications* (pp. 90-100). New Orleans: WMCSA.
- Perkins, C., Belding-Royer, E., & Das, S. (2003). *Ad hoc On-Demand Distance Vector (AODV) Routing*. The Internet Society.

Ramaswamy, S., Fu, H., & Nygard, K. (2005). Simulation Study of Multiple Black Hole Attack on Mobile Ad Hoc Networks. *International Conference on Wireless Networks*, (pp. 595-604). Las Vegas.

Ramaswamy, S., Huirong, F., Manohar, S., Dixon, J., & Nygard, K. (2003). Prevention of Cooperative Black Hole in Wireless Ad Hoc Networks. *International Conference on Wireless Networks*. Las Vegas.

Weerasinghe, H., & Fu, H. (2007). Preventing Cooperative Black Hole Attacks in Mobile Ad Hoc Networks: Simulation Implementation and Evaluation. *Future Generation Communication and Networking*. 1, pp. 362-367. Jeju Island: IEEE Computer Society.

Wu, B., Chen, J., Wu, J., & Cardei, M. (2007). A Survey on Attacks and counter measures in Mobile Ad Hoc Networks. *Signals and Communication Technology* (pp. 103-135). Springer Science+Business Media.