PROTOCOLS FOR DETECTION AND REMOVAL OF WORMHOLES FOR SECURE ROUTING AND NEIGHBORHOOD CREATION IN WIRELESS AD HOC NETWORKS

by

Thaier Saleh Hayajneh

BSc EE, Jordan University of Science & Technology, 1997

MS ECE, Jordan University of Science & Technology, 1999

MS Telecommunication, University of Pittsburgh, 2005

Submitted to the Graduate Faculty of

the School of Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2009

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This dissertation was presented

by

Thaier Saleh Hayajneh

It was defended on

July 28 2009

and approved by

Dr. Prashant Krishnamurthy, Associate Professor, SIS, University of Pittsburgh

Dr. David Tipper, Associate Professor, SIS, University of Pittsburgh

Dr. Richard Thompson, Professor, SIS, University of Pittsburgh

Dr. James Joshi, Associate Professor, SIS, University of Pittsburgh

Dr. Mohamed Eltoweissy, Associate Professor, ECE, Virginia Tech

Dissertation Director: Dr. Prashant Krishnamurthy, Associate Professor, SIS, University of Pittsburgh

PROTOCOLS FOR DETECTION AND REMOVAL OF WORMHOLES FOR SECURE ROUTING AND NEIGHBORHOOD CREATION IN WIRELESS AD HOC NETWORKS

Thaier Saleh Hayajneh, PhD

University of Pittsburgh, 2009

Wireless ad hoc networks are suitable and sometimes the only solution for several applications. Many applications, particularly those in military and critical civilian domains (such as battlefield surveillance and emergency rescue) require that ad hoc networks be secure and stable. In fact, security is one of the main barriers to the extensive use of ad hoc networks in many operations. The primary objective of this dissertation is to propose protocols which will protect ad hoc networks from wormhole attacks - one of the most devastating security attacks - and to improve network stability. Protocols that depend solely on cryptography techniques such as authentication and encryption can prevent/detect several types of security attacks; however, they will not be able to detect or prevent a wormhole attack. This attack on routing in ad hoc networks is also considered to be the main threat against neighborhood discovery protocols. Most of the proposed mechanisms designed to defend against this type of attack are based on location information or time measurements, or require additional hardware or a central entity. Other protocols that relied on connectivity or neighborhood information cannot successfully detect all of the various types and cases of wormhole attacks. In the first part of this dissertation, we present a simple, yet effective protocol to detect wormhole attacks along routes in ad hoc networks. The protocol is evaluated using analysis and simulations. In the second part, we present a secure neighbor creation protocol that can securely discover the neighbors of a node in ad hoc networks, and detect and remove wormhole links, if they exist. The proposed protocols do not require any location information, time synchronization, or special hardware to detect wormhole attacks. To the best of our knowledge, this is the first protocol that makes use of cooperation rules between honest nodes. Use of such rules will reduce the overhead associated with the number of checks to be performed in order to detect wormholes and to create a secure neighborhood. This is also the first protocol, to our knowledge, that addresses the complete removal of bogus links without removing legal links.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

All my prayers are for Allah, the Almighty, who gave me the strength and the patience to accomplish this work and to successfully reach this stage in my life. There are not words enough to thank my advisor, Dr. Prashant Krishnamurthy, whose guidance and support were the cure for several obstacles in preparing, presenting, and writing this dissertation. I am also deeply thankful to my co-advisor, Dr. David Tipper, and all the other committee members: Dr.Richard Thompson, Dr. James Joshi, and Dr. Mohamed Eltoweissy for their cooperation as well as all of their valuable comments and suggestions.

Then, I am thankful for my father (Saleh Hayajneh) who was the main inspiration for me to to pursue my Ph.D. Ever since I was a child he was always telling me that a Ph.D. will put you at the pinnacle of your profession; it was always his dream to see me a Doctor one day. I am also thankful for the support and love that I got from my family members: my mother (Ibtisam), my faithful brother (Abdullah), my sweet sister (Wafa), my step mother (Lieve), and my brother in law (Dr.Mohammed).

Finally, I am thankful for all my friends and the people who loved and supported me throughout my Ph.D studies at the School of Information Sciences.

1.0 INTRODUCTION

Ad hoc is originally a Latin word that literally means “for this purpose only”. Thus, (in telecommunication) it refers usually to temporary networks that are constructed on the fly for some special applications. The nodes in an ad hoc network are simply deployed in an area without any a priori planning. The nodes are capable of organizing themselves, by discovering their neighbors and communicating over the wireless medium. Recently, ad hoc networks have been getting greater attention as more applications are depending on them. Researchers have tried to propose protocols that will improve the quality of service for ad hoc networks in the wireless environments. Many of the applications, particularly military applications, require a high level of security. Thus, the main challenge is to protect ad hoc networks from security attacks. Ad hoc networks use the open wireless medium to communicate, making it easy for an attacker to launch his attacks by injecting, blocking, or modifying the packets. All the nodes act as routers for the data packets and there is no clear line of defense where it is possible to place a firewall-like device. We elaborate on possible security attacks in Chapter 2.

In ad hoc networks one of the most challenging attacks to defend against is the wormhole attack. In a sophisticated version of this attack, the attacker will copy all the control and selected data packets from one location in the network and almost simultaneously replay them at another location in the network. The attacker’s nodes do not need to share any credentials (IDs or keys) with nodes in the network as they can just copy the encrypted or signed packets and replay them. The two locations will be several hops away from each other, but will be connected through a high-speed wired or wireless link controlled by the attacker. Obviously, the goal of the attacker is **not** to improve the network connectivity by helping connecting far away nodes, but to draw traffic through the wormhole.

The wormhole attack can be launched regardless of the MAC, routing, or cryptographic protocols used in the network and is thus difficult to defend against. Defense mechanisms against this attack are either very complex or very expensive. Most of the wormhole defense mechanisms aim to detect wormholes successfully with minimal false positives. Unfortunately, the defense schemes ignore the removal of the links created by the wormhole. We note that a single two-end wormhole could be thought of logically as a single link. In reality, the wormhole creates a large number of links between many nodes in the network. The nodes will not be aware of this fact and will be using the wormhole links as legal links. In Chapter 2, we will discuss the potentially catastrophic impacts of the wormhole attack in more detail.

1.1 PROBLEM NARRATIVE

In this dissertation, we look at two ways in which wormholes will impact network effectivnes and stability.

First, during the route discovery process, a wormhole can relay route request and response messages between distant nodes, creating the appearance of shorter routes to destinations. Since the wormhole can be anywhere along a route, a source will have to detect its existence *somewhere along the route* when a node sets up the route (on-demand). Many researchers have proposed protocols that require special hardware to detect wormholes. For example, suggestions have included the use of directional or smart antennas [1], ultrasound [2], or special RF devices [3]. Other existing protocols require accurate timing and synchronized clocks, accurate location information, or global topology and connectivity information [4, 5, 6]. They all have their advantages, disadvantages, and limitations that will be discussed in Chapter 2.

Second, neighbor discovery is a procedure used by the nodes in ad hoc networks to discover the other nodes with whom they can communicate directly. Neighbor discovery is one of the essential operations in ad hoc networks, in that many of the networking operations in ad hoc networks depend on successful and correct neighbor discovery. Such a need arises, for example, with location-based applications that require nodes to know all nodes in their vicinity, and *1*, *2*, or even k-hop neighbors. Compared to wired networks, the neighbor discovery in wireless networks is more complicated. The nodes in mobile ad hoc networks could be mobile and the list of neighbors of any node keeps changing. Also, wireless links are prone to failure and lack the the stability that exists in wired links. Thus, dynamic changes are possible in the neighborhood making it difficult to identify whether a wormhole is corrupting the neighbor discovery process. A detailed description of the available protocols for secure neighbor discovery in ad hoc networks is presented in Chapter 2.

We elaborate on neighbor discovery below to clarify the differences between on-demand wormhole detection and periodic complete secure neighborhood creation. In its simplest form, neighbor discovery could be done by making the nodes broadcast a simple "hello" message and wait for the reply from their neighbors. Because of its importance to the ad hoc networking functionality, neighbor discovery is a tempting target for an adversary that wants to disrupt the network operation. A simple attack could be to jam the neighbor discovery "hello" message or the reply message. This will prevent two neighbors from discovering each other and could have an impact on networking operations, such as routing. With jamming attacks, the attacker will be broadcasting a jamming signal and it is most likely that the nodes around the jammer will sense some change in the physical characteristics of the received signals. Thus, the nodes may be able to detect the problem and respond with some action (moving from the jammed area, or using a frequency hopping spread spectrum). The impact, detection, and prevention of jamming attacks have been addressed in the literature [7] and will not be considered in this dissertation.

Instead of preventing actual neighbors from discovering each other, an adversary could commit a more disruptive action by making nodes that are far away believe that they are neighbors. This could be achieved by impersonating the identity of other nodes and spoofing fake messages on their behalf. This malicious action could be simply prevented by using cryptographic techniques. The nodes can encrypt or digitally sign the exchanged "hello" discovery messages. Such approaches have been comprehensively covered for wired networks and suitable cryptographic methods, including public key cryptography, that take into account the limitations of ad hoc networks have been presented [8]. Hence, they will not be discussed in this dissertation.

The malicious action of interest to us, that an attacker could perform, to disrupt the neighbor discovery operation, is the wormhole attack. As we explained earlier, here the adversary can copy all the discovery "hello" signals and almost instantly replay them from another location. Nodes at a distant location will receive the "hello" messages and reply to them. The replies will be copied and replayed at the original location. Now the nodes at both locations will mistakenly believe that they are actual neighbors located within range of each other. This attack is much harder to detect compared to the previous ones as it will not cause any change to the physical layer characteristics. Moreover, it is impossible to detect using cryptography, as even encrypted or digitally signed signals can still be copied and replayed. The only way to stop the wormhole attack on neighbor discovery is to somehow discover which nodes are connected via a bogus link and notify the nodes that are bogusly connected about this fact. The nodes can then discard all messages delivered to them from fake neighbors and also not consider them in routing or other operations. In other words, the network has to be sanitized from the fake links created by the wormhole. It is also worth mentioning that the attacker can also control the number of nodes he wants to connect. Thus, he can connect only a single or limited number of nodes at each side, causing minimal noticeable change in the network topology. This makes the protocols that look for changes in the network topology or connectivity structure less than effective.

For secure neighborhood creation, every pair of nodes will have to ensure that they are *real* neighbors. In a large and dense network with too many nodes and large average node degrees it will obviously be too expensive for each node to check whether all its neighbors are real or connected via a wormhole. The problem will even be worse if the nodes are mobile and the network is dynamic (where the neighbors of any node keep changing with time). If the goal of the security mechanism is to simply detect the existence of wormholes that may be attracting traffic, using a wormhole detection protocol on-demand would be a better choice. This means that only a source node that wants to communicate with a destination node will check if a wormhole exists along the chosen route. If, however, the goal of the security mechanism is to ensure that *every* node finds its real neighbors, there is a need for a secure neighbor creation protocol that can provide a complete check for the entire network and verify if all the neighbors of all the nodes are real and not connected through a wormhole. Thus, we first present an on-demand wormhole detection mechanism and then consider the problem of secure neighborhood creation.

There are many aspects that must be taken into account in either of the two problems consideraed in this dissertation. In what follows, we discuss some of the most important metrics as well as our recommendations and justifications related to them. More details, explanations, and examples will presented throughout the dissertation. While we evaluate some of these metrics explicitly in the dissertation, others are qualitatively analyzed and arguments as to the appropriateness of the proposed solutions are presented.

- *Speed*: The protocol must operate quickly. This will make it less likely that the situation or dynamics in the network will change. Moreover, it is not preferable to have the protocol cause delay for applications that need to know the neighborhood information of nodes in the network.
- *Cost*: The protocol must not require a large overhead in terms of communication and computation. The protocol must not result in exchanging a large volume of packets in order to discover the neighbors. This may clog the network and cause delay for other network functionalities and protocols. It is also preferable that the protocol not require the addition of large pieces of information to every packet (e.g. packet leashes [9]).
- *Centralization*: It is always recommended that the protocol be decentralized and, ideally, completely distributed. Thus, the protocol should not require a central entity, which is a significant advantage [10]. The protocol should be localized and only rely on information that can be collected by the nodes or their immediate neighbors.
- *Requirements*: If possible the protocol should not rely on any one of the following:
  - Additional hardware: Many protocols require additional hardware such as: ultrasound, special RF, ...etc. This makes them difficult to apply to every network and require special customization.
  - Time Measurements: Protocols that are based on accurate time measurements cannot detect all types of wormholes [11]. This will be discussed in detail in Chapter 2.

  - Synchronized clocks: Many protocols require the nodes to have accurately synchronized clocks. Which requires a special protocol for synchronization, such a protocol makes them vulnerable to clock synchronization attacks. In addition, it cannot detect all types of wormholes.

  - Location information: Many protocols require the nodes to have accurate location information. This requires the nodes to be equipped with GPS or to have some localization protocol. These protocols are hard to implement and not accurate, especially in indoor and urban areas.

- *Connectivity*: The protocol must not require that the nodes in the network be uniformly distributed or static with fixed edges. In mobile ad hoc networks, the nodes may be randomly distributed and may have dynamic links. Moreover, the protocol must not require that the nodes have large node degrees to function properly.

- *Detection*: The protocol must be capable of: (i) successfully detecting various types of wormholes. (ii) accounting for unique topological situations, and (iii) working for all types of wormholes without special constraints. For example, the protocol should not depend on the number of nodes connected through the wormhole. In addition, the protocol should not require the connected nodes to form any special topology structure.

- *False positives*: The protocol must produce a very low percentage of false positives, if any. This is because the actions that could be taken if a wormhole is falsely detected could have a negative impact on network operation.

- *Wormhole removal*: Detection of wormholes is not enough – ideally, the protocol must also successfully remove all the bogus links created by the wormhole. If a single link is not removed, it most likely means that the removal process has failed. Moreover, the protocol must not mistakenly remove too many legal links in the network as this may disrupt the network topology or isolate some nodes.

1.2 ORGANIZATION OF THE DISSERTATION

Chapter 2 of this dissertation will present the background material. We start with a brief introduction about ad hoc networks, some definitions, applications, and protocols used at each layer. Security attacks at various layers in ad hoc networks are discussed. The wormhole attack is presented in further details, exploring its definition, impact, and types. Neighbor discovery in ad hoc networks is defined. A general classification of the approaches used to detect wormhole attacks is presented. The available detection protocols and secure neighbor discovery protocols are presented and classified.

In Chapter 3 we present, "DeWorm", a novel on-demand protocol to detect wormhole attacks in ad hoc networks. Here, we note that for a wormhole attack to successfully interrupt the network, it must attract a significant amount of network traffic by providing a perceived short-cut through the network. Hence, routes going through the wormhole must be shorter than alternate routes through valid network nodes. Exploiting this observation is the basis of our wormhole detection protocol "DeWorm". Specifically, in DeWorm, we detect wormhole attacks by routing hop-count discrepancies between neighboring nodes along a path from a source to a destination. The proposed protocol is simple and localized, can be applied on demand (when the existence or lack thereof of a wormhole needs to be verified), needs no special hardware, localization, or synchronization, and can detect physical layer wormholes. Unlike other protocols that rely on connectivity or neighborhood information (described in Chapter 2) DeWorm can detect wormholes without requiring that the nodes be uniformly and densely deployed. The protocol is explained with details and examples. Several issues related to the protocol will be analyzed and discussed. We present simulation-based results evaluating DeWorm in grid-like and randomly distributed networks with various connectivity models (unit disk graph - UDG, Quasi-UDG, both with symmetric and asymmetric links). Our numerical results show that DeWorm has a high wormhole detection rate and few false positives. The cost of the protocol is the addition of a slight amount of overhead and usually when executed in on-demand fashion for wormhole detection.

Chapter 4 will present the proposed secure neighborhood creation protocol (developed in this dissertation) which can securely discover the neighbors of nodes in a mobile ad hoc network, detecting and removing any existing wormhole links. Compared to other secure neighbor verification or discovery protocols, our protocol is simple, localized, needs no special hardware, localization, or synchronization. The protocol can also detect and remove two-ended and multi-ended wormholes. The proposed secure neighbor creation protocol consists of two main processes: the *detection* process and the *removal* process. The first process, the detection process, includes three operations: initial detection, mutual detection, and co-operation and control. The initial detection operation is a modified version of DeWorm. It provides almost 100% detection for various types of wormholes, but unfortunately results in a high number of false positives. The mutual detection operation will enhance the performance of the initial detection operation by maintaining the high detection rates while significantly reducing the number of false positives. The co-operation and control operation manages the use of the previous two stages (e.g., instead of having all nodes applying the initial detection process to links with all of their purported neighbors, this stage will define the rules that each node will follow) reducing the overall overhead by almost 80%. This process will still maintain almost 100% detection and very few numbers of false positives. The removal process consists of two operations: initial removal and mutual removal. This process is the most complicated stage and depends on the results produced by the previous stages. If there are no wormholes that exist in the network, ideally, the removal process is not supposed to remove any link by mistake. However, if there is a wormhole, then it must successfully remove all bogus links between nodes connected using the wormhole. The mutual removal operation will enhance the performance of the initial removal operation by reducing the number of links removed by mistake. The simulations and analysis show that the proposed protocol can successfully detect and remove all wormhole links. Very few false positives will occur and very few legal links will be removed by mistake. The cost, as will be evaluated, is very low overhead measured by the number of route acquisitions. To our knowledge, this the first protocol that employs co-operation between neighbors to reduce the overhead that may be incurred if all nodes in the network need to verify links to all of their neighbors. Also, this is the first protocol that can provide near 100% removal of all the wormhole based links and remove only a very few, if any, legal links.

Chapter 5 will conclude this dissertation and discuss the future work that includes issues that we would like to pursue and complete in our future research.

1.3 CONTRIBUTIONS

The main contributions of this dissertation are as follows:

- We have proposed and evaluated "DeWorm", a novel and simple protocol that does not require any location information, time synchronization, or special hardware to detect wormhole attacks. This protocol has a very high detection rate and low false positive rate, and can be applied to almost all types of wormholes and network topologies.
- We have proposed simple modifications to detect some advanced wormhole attacks that may bypass DeWorm.
- We have proposed a secure neighborhood creation protocol in the face of wormhole attacks, based in principle on the DeWorm protocol. The new protocol not only detects wormholes, but also identifies and removes bogus links.
- We have employed a novel co-operation and control operation that will reduce the overhead of the neighbor creation protocol by up to 80% as compared to brute force checks between all pairs of links.

2.0 BACKGROUND

2.1 AD HOC NETWORKS

2.1.1 Definitions and Applications

An ad hoc network is a collection of nodes that communicate with each other through wireless links and without the presence (or need) of any fixed infrastructure. Such properties make it challenging to design suitable protocols that must be aware of the system requirements. Nodes in ad hoc network have to be able to automatically discover their neighbors and cooperate together to send data packets from a source node to a destination node. Nodes that are within each others transmission range can communicate directly (referred to as single-hop communications). However, nodes that are not within each others transmission range have to depend on other nodes to deliver their packets (referred to as multi-hop communications). The discovery of the neighbors of nodes and routes from one node to another are activities that have to be performed by all of the nodes locally without assistance from any central entity.

Nodes may vary in size and capabilities. For example, they could be tiny small sensors with very limited computation, communication, and energy capabilities. Thus, they should use protocols that are aware of these limitations. On the other hand, larger more powerful nodes such as laptops or even vehicles that are equipped with communication and computation devices are also possible. Nodes may be deployed in large numbers and in a large area. The nodes could be distributed in the network either randomly or in a fixed grid. In this dissertation we assume nodes are homogeneous, but make no specific assumptions about their limitations (e.g., we do not assume that nodes are tiny sensors).

The ability to construct networks instantaneously without wiring or a priori design or planning makes ad hoc networks attractive for many applications. They are the only choice in many military applications – for example, soldiers invading a country may not be able to use the local operator or have the chance to build an infrastructure. Moreover, the Wireless Emergency Rescue Team **ADD CITATION** recommended that telecom operators should provide an ad hoc mode for their infrastructure in order to operate in emergency situations for co-operation with police, firemen and hospital services. Ad hoc networks are also be useful in business meetings where participants need to share information during the meeting and also in interactive conferences where attendees need to use their laptops to participate in the discussion. These applications are besides many other industrial, personal, and home applications. For example, they could be used for machine monitoring in manufacturing plants, in a smart home environment, and for green-energy based buildings.

2.1.2 Protocols

The most commonly used technologies for ad hoc networks are IEEE 802.11 and Bluetooth. In this dissertation we will consider IEEE 802.11 networks operating near a center frequency of 2.4 GHz. This is an open unlicensed frequency band that is shared by other technologies such as Bluetooth, and Zigbee. Thus, there may be interference that could effect the performance of the network, this is referred to as the co-existence problem [12]. However, interference is not an issue considered in this dissertation.

The medium access control (MAC) protocols in ad hoc networks are mainly classified as: contention-free and contention-based MAC protocols [13]. With contention-free MAC protocols, the nodes do not compete to access the medium, mechanisms such as: TDMA, CDMA, or FDMA are used where the nodes will share the time, code, or frequency, respectively. These MAC protocols allow a fair share of the medium and guarantee a bounded delay and specific bandwidth per node. However, they are more complex to manage and maintain, and may require synchronization or centralized management. Contention-based MAC protocols are mainly based on Carrier Sense Multiple Access (CSMA) scheme. These are most common and practical in ad hoc networks as the topologies can change temporally and spatially.

Routing protocols are very important in ad hoc networks as they allow nodes to communicate with other nodes that are farther than one-hop away. However, designing routing protocols is challenging due to fact that ad hoc networks by definition do not have a fixed and known infrastructure. Routing protocols are generally classified into topology-based and position-based protocols. A comprehensive survey of routing protocols for ad hoc networks is provided in [14]. Position-based protocols are not preferred in ad hoc networks as they require the nodes to know their geographical location. This requires running a complicated localization protocol or for all nodes to be equipped with GPS, which may still not provide the required accuracy. Topology-based routing protocols are further classified into proactive and reactive protocols.

Proactive routing is similar to traditional routing protocols used in the Internet. The routing information is collected and shared even if there are no routing requests. The routing information has to be updated periodically or when any change occurs in the network topology. Due to mobility and frequent link and node failures in ad hoc networks the network topology is expected to change frequently. Thus, proactive routing is also not favored as it will require frequent update messages resulting in a large communication overhead. Examples of proactive routing protocols include Destination Sequenced Distance Vector (DSDV) which is similar to the Bellman-Ford algorithm and Optimal Link State Routing (OLSR) which is based on link state algorithm.

In the case of DSDV, routing tables are maintained at each node and used to make packet forwarding decisions. The protocol adds a sequence number to each route table entry in each node. In order to maintain the consistency of the routing tables in a changing environment, each station transmits updates of its routing table periodically by using broadcasting or multicasting. The route advertisements from a node indicate which nodes are accessible and how many hops are required to reach that particular node in order to ensure the shortest number of hops for reaching a destination. The packets may be transmitted so that they contain layer 2 or layer 3 addresses.

Reactive routing is also referred to as on-demand routing. As the name implies, these protocols only operate when there is an explicit routing request. The node that wants to know a route to a destination node will initiate a route request. This process will end when a node discovers its requested route or all the possible routes have been examined without finding a route. The most common reactive routing protocols are: Ad hoc On-demand Distance Vector (AODV) which is also based on the Bellman-Ford algorithm and Dynamic Source Routing (DSR) in which each node will maintain a route cache of the source routes it knows.

The DSR protocol allows the discovery and maintenance of routes in ad hoc networks by using two mechanisms: route discovery and route maintenance.The DSR protocol does not require any periodic packets to be broadcast within the network. When a node S wants to to find a route to node D for the first time, it will initiate a route discovery protocol. S will locally broadcast a Route Request message. This message indicates S as the initiator, D as the target, and a unique request ID. It will be received by all the one hop neighbors of S and each node that receives it, for the first time, will add itself to the route record and broadcast the message locally. If the target node D receivers the route request message it will send a Route Reply message to node S. The message will include the accumulative route record. Node S, after receiving the a Route Reply packet, will cache the route in its Route Cache and use it to send subsequent packets. If a node that is not the target receives another Route Request message from the same initiater with the same ID, or finds that it is listed in the route record, it discards the message. The route discovery will be performed by a sender S who wishes to send a packet to the destination D, only if S does not know a route to D. Route maintenance will allow S to detect if the links in the route it already has to D are no longer available. This could be due to topology changes that caused the old route to be broken. In this case S can run a new route discovery process to find a working route. The route maintenance will be used only when S is sending packets.

2.2 SECURITY ISSUES IN AD HOC NETWORKS

It is not enough to have an ad hoc network work with acceptable level of reliability and quality of service. It is also important to provide an adequate level of security. Many of the aforementioned applications are critical and cannot be deployed without granting a certain level of security. For example in military applications, it is important that an adversary not be able to listen to the commands that are sent to the soldier, not be able to inject fake commands, nor replay legitimate commands. For applications that grant access for nodes to certain resources based on the nodes' locations, it is crucial not to allow anyone to fake his location and grant forbidden access. There are many such issues that have to be taken into account when a security protocol is designed for ad hoc networks.

There are several reasons that make security in ad hoc networks different and more challenging than wired networks. First, in ad hoc networks, the nodes use the wireless medium to communicate with each other. Thus it is easy for an adversary to eavesdrop, modify, or inject false packets as the medium is open and the attacker does not have to physically tap into network wires to gain access. Moreover, in ad hoc networks there is no clear line of defense compared to wired networks where one can place his firewalls or gateways at the ingress to the network to prevent illegitimate access to the network. In addition, nodes in ad hoc networks also act as routers and are required to forward packets sent from their neighbors in a multi-hop manner. Thus a selfish or malicious node can choose to drop and not forward packet in order to save its energy or disrupt the network operation.

In general the main security requirements for any system will be: confidentiality, authentication, integrity, non-repudiation, availability, and access control. Confidentiality ensures that eavesdroppers will not be able to read the information sent through the network which may be achieved by encrypting data and control packets. Authentication prevents impersonation and verifies the identity of the nodes. Integrity will insure that packets will not be modified or altered by an adversary. Non-repudiation prevents a node from denying that it has sent a message after it does so. Availability implies that the network services must be available to legitimate users regardless of any malicious incidents. Finally, access control will set the rules that specify the rights and permissions that are granted for each node.

There are many different aspects to consider in order to classify attacks in ad hoc networks [15]. They can be classified into passive and active attacks depending on how much the attacker is involved. In passive attacks the attacker will only do things like eavesdropping, traffic analysis, and monitoring. In active attacks the attacker will be more involved and may jam, modify, spoof or replay packets. Another classification depends on the domain of the attack. This classifies attacks into internal and external attacks or insider and outsider attacks. Internal attacks come from compromised nodes who already share cryptographic keys with other nodes and participate in the network operation. Usually, these attacks are harder to detect and can cause more damage to the network compared to the external attacks that are performed by nodes that are not part of the network. Depending on whether the attacker tries to hide his malicious action from being detected or not, attacks can be classified to stealthy or non-stealthy attacks. Moreover, attacks can be classified to cryptography or non-cryptography related attacks. In cryptography related attacks the attacker is trying to break the cryptographic protocols or guessing the secrets used in the protocol, for example, using brute force attack to find the key used in an encryption system. In non-cryptography related attacks the attacker will try to make use of the faults in protocol design without breaking the encryption system (or bypassing it entirely). Finally, the most common classification used in the literature is to classify the attacks according to the five layers of the Internet model. Examples of some attacks at each layer are shown table 2.1 [15].

Layer	Examples of Security Attacks
Physical layer	Jamming, interference, eavesdropping
Data link layer	MAC misbehaving, traffic analysis
Network layer	Wormhole, blackhole, incorrect traffic generation
Transport layer	SYN flooding, session hijacking
Application layer	Data corruption, repudiation

Table 2.1: Examples of security attacks at each layer

At the physical layer, jamming attacks are considered one of the most serious attacks in ad hoc networks [7]. Jamming attacks happen when an attacker interferes with the radio frequencies used by the network nodes. Constant jamming could prevent the nodes from using the network causing denial-of-service (DoS), and it could also prevent the nodes from reporting the attack. Different forms of spread spectrum communications are used as a typical defense against jamming. Jamming is still possible in this case, but more difficult since an attacker needs to follow the exact hopping sequence or to jam a wide section of the band. In [16], researchers have studied the impact of limited-range jamming attacks on ad hoc networks. Limited-range jammers use low power (close to regular node power) to jam small regions in the network. Hence, these jammers are much harder to detect compared to high power jammers.

MAC misbehaving is one of the common attacks at the MAC layer. In this attack the nodes will not follow the rules of the MAC protocol. As we have discussed earlier, the MAC layer in ad hoc networks is typically based on CSMA/CA, which means that the nodes will sense the channel and will only transmit if the medium is free. Thus, a malicious node may start transmitting without waiting and not using back-offs causing the other nodes to continually back off. Researchers have extensively addressed this problem and proposed some prevention techniques [17].

Location obfuscation is another issue that needs to be considered in some applications. The location of communicating entities in wireless ad hoc networks is extremely important due to the potential of their being identified and subsequently subjected to attacks (e.g., in military networks). In [18] researchers showed that analysis of traffic in ad hoc networks may reveal the locations of command centers enabling the adversaries to launch targeted cyber or physical attacks on them. Hence, it is more critical to hide the location of the source as well as ensure the seclusion of destination for quasi-stationary nodes in ad hoc networks. Different methods have been proposed in the literature for defending against traffic analysis and location identification attacks. The researchers in [18] proposed SECLOUD: Source and Destination Seclusion using Clouds to obfuscate the true source/destination nodes and make them indistinguishable among a group of neighbor nodes which works well even under network-wide traffic visualization by a global attacker.

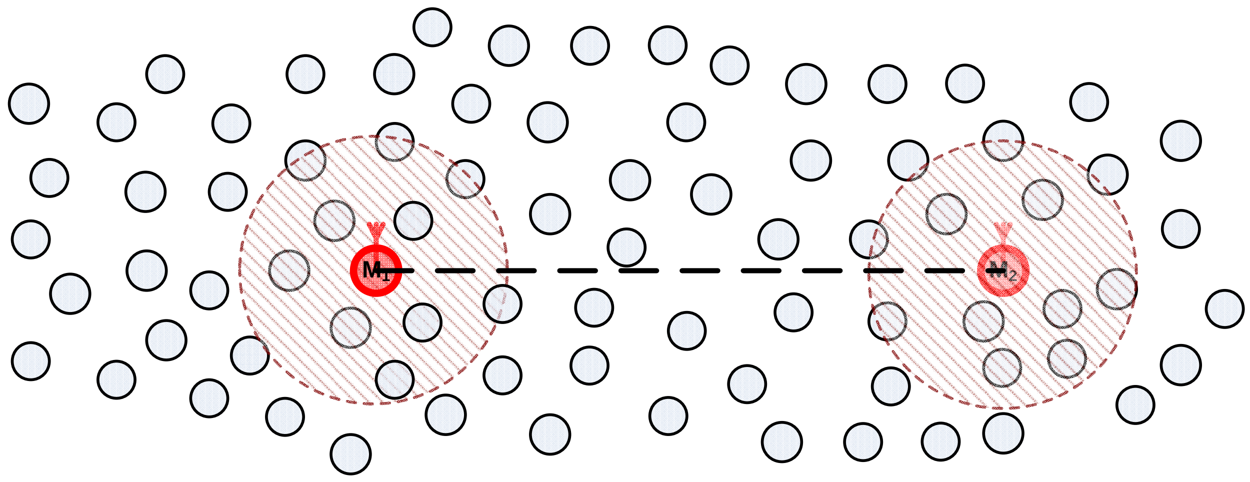

Figure 2.1: Wormhole Attack

2.3 WORMHOLE ATTACKS

2.3.1 Definition

The wormhole attack in ad hoc and sensor networks [9, 19, 20] is impervious to traditional security measures. Wormhole attacks can be launched regardless of the MAC, routing, or security protocol used in the network. Here, an attacker will place two transceivers M_1 and M_2 at two physically different locations in the network as shown in Figure 2.1. The transceivers M_1 and M_2 are connected through a wired or long range wireless link called *the wormhole link*. These transceivers capture packets or signals from one location and replay them at the other location. Alternatively, regular nodes controlled by an attacker can be used to tunnel packets from $M - 1$ to M_2. Legitimate nodes consider the wormhole link as a short path from one side of the network to the other side (e.g., nodes at M_1 location in Figure 2.1 will assume that nodes at M_2 location are one-hop neighbors). Encryption and authentication do not help as the transceivers simply relay the encrypted or authenticated packets or signals.

Consequently, the wormhole will attract a large amount of traffic between various source and destination nodes in the network. For example, researchers in [21] showed that strategic placement of a wormhole, in a network where the nodes are uniformly and independently distributed, can impact 32% of all communications in the network, on average. The nodes at M_1 location in Figure 2.1 and the surrounding nodes including all the nodes located to the left of M_1 will most most likely use the wormhole link to reach the nodes located at or to the right of M_2 location.

2.3.2 Impacts of wormhole attacks

If the wormhole will only peacefully transport all the traffic from one location in the network to another location that is far away, then it could be useful for the network operation as it will improve the network connectivity. Unfortunately, once the traffic is routed through the wormhole, the attacker will gain full control over the traffic. Then he can start his malicious actions by selectively dropping data packets which will lower the network throughput or store all the traffic and later perform cryptanalysis attacks. The attacker can decide when to drop data packets that passes through the wormhole at some critical situations. For example, if the network is used for some alarm or surveillance system, then the attacker can decide to time his packet dropping with a planned intrusion into the system. The wormhole attack was shown to have significant impact on both proactive and reactive ad hoc routing protocols [22]. Turning the wormhole link off suddenly means that all the nodes that used the wormhole to reach other nodes will have to find new routes. Thus, the network will be clogged with many route requests disrupting the operation of the network leading to DoS attack. The attacker will make use of this situation and may periodically turning the wormhole link off and on. Furthermore, the attacker can turn a node to become a sinkhole [23] and may cause that node to be mistakenly blacklisted by other nodes in the network.

In addition wormhole attacks will easily disorder localization protocols [24, 25] or deceive protocols that manage access control depending on the geographic location of the nodes. In this case the attacker will establish a wormhole link between one location that is managed by an access server to another location and grant access to unauthorized nodes. Using complicated protocols that include time stamps, certifications, or reliance on a trusted third party will not effectively detect or prevent the wormhole attack.

2.3.3 Types of Wormhole Attacks

Wormhole attacks were classified based on the type of links used by M_1 and M_2 (existing wireless data paths - in-band, or high speed channels - out-of-band) [26, 27]. With in-band wormholes usually the attackers are insider nodes that use the same radio channel used by the other nodes in the network. The nodes will try to increase their transmission range by transmitting at the highest possible power and will not follow the MAC protocol waiting times to ensure faster delivery. Such attacks are also referred to as rushing attacks [28] and are considered to be easier to launch. On the other hand, in out-of-band attacks the attacker will connect his nodes with long range fast connections and this can be either a long range wireless link that uses a different radio frequency (compared to the bands used by the ad hoc network) or a fast wired link. Out-of-band wormholes are more advanced and damaging because the longer and faster the wormhole, the more nodes are attracted to send traffic through it and the more damage and disruption it can cause to the network.

In [29] wormhole attacks were classified according to whether nodes M_1 and M_2 are visible on the route (or simply replay packets). They were classified into three types: closed, half open, and open wormhole attacks. With a closed wormhole neither M_1 nor M_2 are visible to their neighbors, which means they do not advertise their node IDs or MAC addresses. Either M_1 or M_2 will be visible to its neighbors in half-open wormhole, while in open wormholes both M_1 and M_2 will be visible to their neighbors. Closed wormhole attacks are considered much harder to detect compared to open wormhole attacks. This is because with open wormholes, the adversarial nodes are visible to their neighbors and must follow the rules of the IEEE 802.11 MAC protocol. Thus, there will be a *minimum* packet relaying delay that could be used by the neighboring nodes to detect the attack. Also any violation to the MAC rules could be detected by the nodes. On the contrary, with closed wormholes, there could be no packet relaying delay caused by the wormhole and it is not necessary to follow the MAC protocol rules. Hence, it is very hard to detect the wormhole with any protocol that relies on time measurements.

Another type of wormhole attack, which is hard to detect, is the physical layer wormhole [30]. In this type the wormhole captures the bits or the waveforms from one side, transmits them on the other side using high speed fiber links, and replays the exact bits or the received waveforms using physical layer repeaters. Transmission and replay can start before the receipt of the entire packet. Hence, the wormhole will not cause any noticeable delay that could be used to detect it.

Khalil et al. [31, 32] have classified the wormhole attacks based on the techniques used for launching it. The wormhole attacks were classified as follows:

1. Wormhole using encapsulation: in this type M_1 will encapsulate the packets coming from its neighbors and send them to M_2 using the nodes in the network. M_2 will broadcast the packets to its neighbors after demarshalling them. Note that, the original encapsulated packets are not viewed by the nodes on the route and thus the hop counts on these packet are not increased thus the nodes at M_2 side will think that the nodes at M_1 side are one-hop away.
2. Wormhole using out-of-band channel: this has been defined earlier in this section.
3. Wormhole using high power transmission: this is similar to the out-of-band wormhole where M_1 will use a high transmit power.
4. Wormhole using packet relays: in this case, M_1 will simply amplify and resend the packets received from the sender. This will increase the one-hop neighbor list of the sender.

2.4 NEIGHBOR DISCOVERY PROTOCOLS

Neighbor discovery could be defined as the process by which nodes will know who all their neighbors are [10]. Communication in ad hoc networks is performed in a multi-hop manner. Nodes cannot reach other nodes that are not within their transmission range without the help from other nodes and the first nodes that are contacted are the direct (one-hop) neighbors. Many applications, protocols, and system functionality rely on neighborhood discovery. In mobile ad hoc networks the nodes are mobile and communicate with their neighbors over wireless links the status of which may vary with time. Thus, the list of neighbors of each node may be changed with time as new nodes could be added to the list and others could be removed. Hence, the neighbor discovery protocol must be used frequently to ensure that the neighboring information of the nodes is fresh.

As is the case with other essential network operations, neighbor discovery is a potential target for malicious attacks. Thus, simply sending a neighbor discovery message and waiting for a reply will not guarantee that all the replies will come from real neighbors located within the node's transmission range. In general the attacks against neighbor discovery could fall under two main types. In the first type the attack aims to shrink the neighboring list, by preventing nodes from discovering their neighbors. This could be achieved by jamming the neighbor discovery message or the reply message. In the jamming attack [7], the attacker will transmit a signal that will collide with a legitimate signal and prevent it from being received correctly. Nodes that are close to the jammer may notice the jamming signal and possibly detect the existence of the jammer. Jamming attacks are beyond the scope of this work and will not be considered in this dissertation.

The second type of attacks against neighbor discovery protocols aim to bogusly extend the neighbors' list. This means the inclusion of nodes that are not within the nodes' transmission range. There are two approaches for this attack. The first one is simple and easier to prevent and the second one is complicated and much harder to prevent. In the first approach, the attacker will simply fake replies from nodes that are located far away or do not exist. This will be prevented by securing the neighbor discovery messages and the replies using cryptographic techniques such as: authentication or encryption. In the second approach the attacker will employ a wormhole attack.

One of the main goals of wormhole attacks is to disrupt the neighbor discovery protocols. Thus, secure neighbor creation protocols must guard the network from wormhole attacks. Figure 2.2 shows a network with wormhole attack. This is a single two-ended closed wormhole attack. As shown, the wormhole connects four nodes from M_1's side (A, D, E, and L) with three nodes at M_2's side (B, F, and X). Consequently, twelve bogus links are created and the one-hop neighboring list of seven nodes are messed up and extended. The first task of any secure neighbor discovery is to detect the existence of the wormhole. This could be done by any of the nodes that are within the range of M_1 or M_2 as it tries to check or verify its

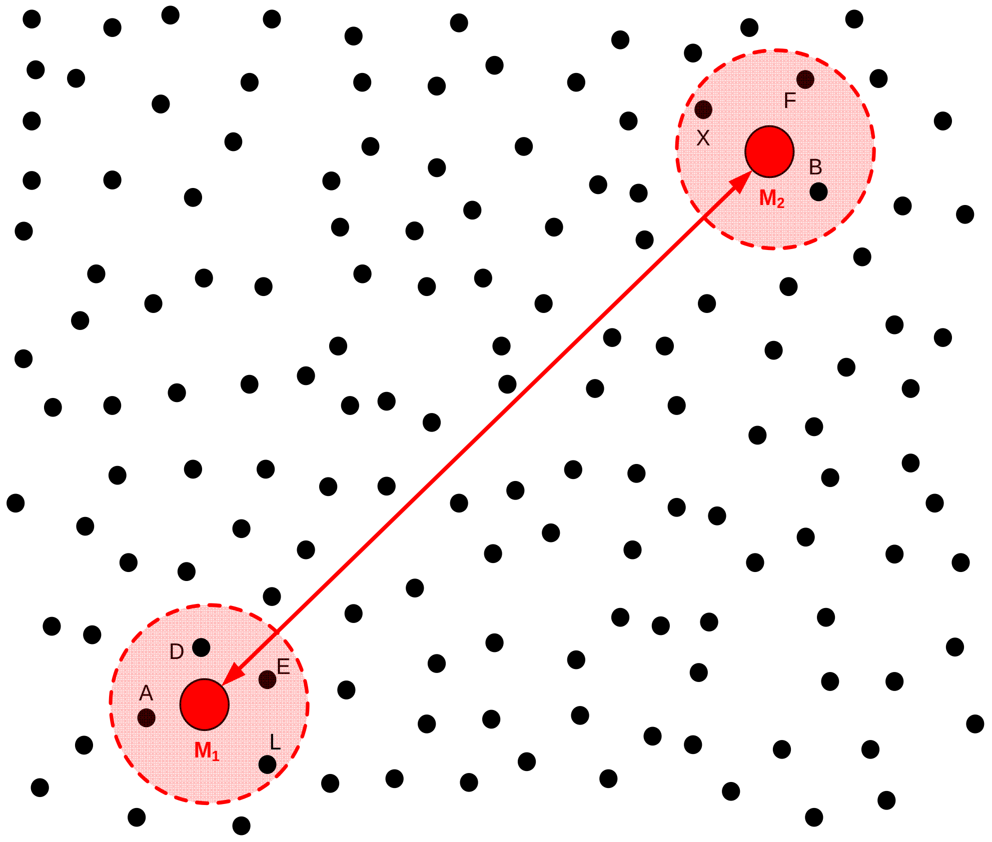

Figure 2.2: Wormhole Against Neighbor Discovery

neighbors.

The detection of the wormhole is not enough to stop the impact of the wormhole. The only way to prevent the wormhole attack is to inform the nodes that are connected via the wormhole about this fact. These nodes must stop sending or forwarding packets to each other. For example, node A must not send or route packets to any of nodes B, F, and X, and it must also discard packets that arrive from these nodes. This process is referred to as *removal* of wormhole links. As will be discussed in details later in Chapter 4, distinguishing between actual and fake neighbors is a complicated operation. The removal of wormhole links must be complete and none of the links should be left active. In the example in Figure 2.2. if all the links were removed except the link between nodes A and B then the wormhole will still be active. This is because the neighboring nodes of node A and B will use this link to communicate. Hence the wormhole will still have the same impact on the network.

There are many aspects that must be taken into account when designing a secure neighbor discovery protocol. In Chapter 1 we have listed some of the most important metrics and our recommendations and justifications of them. In here we will briefly list the main features of a successful secure neighbor discovery protocol.

- The protocol must operate fast so that it can run frequently, if necessary, in the network.
- The protocol must not require large overhead in terms of communication and computation.
- It is always preferred to have the protocol decentralized and distributed.
- It is not preferable to have the protocol rely on one of the following: additional hardware, time measurements, synchronized clocks, or location information.
- The protocol must be capable to successfully detect all types of wormholes including all the special cases.
- The protocol must produce very low percentage of false positives, if any.
- The protocol must also successfully remove all the links created by the wormhole.
- The protocol must not mistakenly remove too many legal links in the network.

2.5 CLASSIFICATION OF SECURE NEIGHBOR DISCOVERY AND WORMHOLE DEFENSE PROTOCOLS

The literature is rich with many defense mechanisms against the wormhole attack and protocols that achieve secure neighbor discovery. There is no clear classification for the protocols in the literature that are commonly prescribed for this purpose. However, researchers have tried to classify the protocols depending on the technology they used to ensure secure neighbor discovery or detect wormhole attacks. For example, the protocols that use time measurements or clocks are referred to as time-based protocols. In this section, based on our review of the literature, we will provide a comprehensive classification for secure neighbor discovery and wormhole defense mechanisms. This classification will be based on the techniques or approaches that are used in the protocols. Many protocols, as will be discussed later in this chapter, have relied on one or more of these techniques. Also many protocols may fall under more than one category. For example, a protocol could be centralized and use time measurements, or require special hardware and use location information. In our discussion of the classes or techniques of the protocols we will also focus on the main challenges with each class. All protocols that use a specific technique will also inherit such challenges, if any, that are involved with using that technique.

Here we will present a general classification for the previously proposed approaches based on the technique they used.

2.5.1 Location-based Approaches

In location-based approaches, nodes in the ad hoc network are assumed to be aware of their locations. The sender and the receiver will securely exchange their location information. Then, to detect whether a wormhole connects them, the nodes will check if packets have traveled the distance between them using only a few hops and/or in a short time. Protocols that depend on location information usually require the nodes to be equipped with GPS, which may not provide the required location accuracy, especially in indoor and urban areas. Some protocols rely on other localization protocols to determine the nodes' locations. However, the accuracy of these protocols is also questionable and they are themselves vulnerable to other attacks [33]. For simplification and cost reduction, some location-based approaches do not require all nodes to be aware of their location, but only a small group must be location aware. This will require few nodes to be equipped with GPS but also requires the network to be somehow clustered. The GPS enabled nodes must be fully distributed and authenticate all nodes in their vicinity, which will be complicated to manage.

2.5.2 Time-based Approaches

Time-based approaches could be sub-classified into two subclasses:

2.5.2.1 Strict time synchronization based approaches These approaches require the nodes to be equipped with clocks that are accurately time synchronized. Some protocols require the accuracy of the clock synchronization to be in the order of few microseconds [9]. Thus, ad hoc networks in this case will need to run an accurate clock synchronization protocol [34]. There are some attacks that target these synchronization protocols – thus they have also to be secured [35]. In these protocols the sender node will attach time stamps to the packets it sends and the receiver will record the arrival time. Then the receiver will check if the packets were delayed longer than some threshold value to detect wormholes. Definitely, the time stamps have to be authenticated to prevent any attempt of illegible modification by an adversarial node.

2.5.2.2 Accurate time measurement based approaches In this case clocks of all the nodes are not required to be synchronized as only one node will be doing the time measurement and responsible for the detection operation. Thus, the sender may send a prob packet or a challenge then starts his timer. The timer will be stopped when the sender receives the response, which is expected to be immediate. Then, depending on some estimations, a threshold value could be used to detect if the packet has passed through a wormhole or not. The assumption is that a wormhole will cause some additional and noticeable delay to the packets that pass through it. Approaches that use time measurements could be effective to detect open wormhole attacks (also typically in-band), where the attacker is using compromised nodes to form the wormhole. In this case all the nodes have to follow the rules of the IEEE 80.11 MAC protocol and it is possible to detect violations. These approaches are very similar to the time synchronization based approaches and they share the same drawbacks.

In general, time-based protocols require some approximations as the node that is in charge of detection has to account for the processing and propagation delay times. Moreover, in ad hoc networks, the MAC protocol may also cause some unpredictable delays. Thus, the threshold value is not expected to be very accurate and the detection process may cause large numbers of false positives. Clock synchronization errors could also cause false positives. More importantly, these protocols are not capable to detect the physical layer wormholes or wormholes that do not cause any delay.

In [36] researchers investigated the available secure neighbor discovery in wireless networks. The protocols were divided into two main categories: time-based protocols and time-and-location-based protocols. They showed that it is impossible to secure the neighborhood with general time-based protocols if adversarial nodes are able to relay messages with a delay below a certain threshold. Note that this threshold is used by these protocols to detect wormholes. On the other hand, they showed that it is possible for time-based protocols to achieve secure neighbor discovery if the minimum relaying delay is above that same threshold. Time-and-location-based protocols are similar to the time-base protocols but with the nodes being aware of their location. These protocols can secure neighbor discovery even if the attacker relay messages with no delay. As will be discussed later, a similar conclusion was also reached by Chiang et. al. [11].

2.5.3 Distance Bounding Approaches

Distance bounding approaches use estimates of the physical distance between purported neighbors to ensure that it is not longer than the maximum allowable distance (e.g., farthest distance reachable by a node operating at its maximum transmission power). Many techniques have been used to estimate the distance between the nodes. Some researchers relied on the signal round trip time and multiplying it by the signal propagation time (speed of light) [37].

These protocols may have the same problem as the time-based approaches. Other approached used ultrasound ranging techniques to approximate the distance between the nodes. This requires the nodes to be equipped with additional hardware. The distance could also be estimated based on the nodes location information or packets time stamps. Thus, some location-based and time-based approaches could also be considered as distance bounding approaches.

2.5.4 Additional Hardware Approaches

Many approaches use some special hardware such as directional antennas [1], special RF [3], or ultrasound [2] to detect wormholes. These protocols cannot be easily applicable to any ad hoc network and they add expense, complexity, and need for special customization. Thus, it is not recommended to propose protocols that rely on additional hardware. Moreover, some of these protocols have their own specific weakness and cannot always ensure the detection of wormholes. Also it is possible for the attacker to use adversarial nodes that are equipped with the same hardware used by the network nodes. For example, an attacker could also use directional antennas and align them in a way to deceive the detection protocol.

2.5.5 Connectivity and Neighborhood-based Approaches

In this case nodes will exchange information about their connectivity such as: node degrees, the list one-hop and/or two-hop neighbors. Then, based on such information, a wormhole could be probabilistically detected. In general, the information should always be locally collected and distributed, that is between a node and its one or at most two-hop neighbors.

The assumption here is that adding a wormhole to the network will cause changes to the connectivity graph or the topology structure of the nodes. Some approaches require the nodes in the network to be uniformly distributed or static with fixed edges. They assumed that adding wormholes must cause a noticeable increase in the node degree of some neighbors. However, in mobile ad hoc networks the nodes could be randomly distributed and could have dynamic links. Also the node degree could vary significantly between nodes. Moreover, some protocols require the nodes to have large node degrees to function properly. In general, the main advantage of the approaches that are based on connectivity of neighbor information is that they do not require any time or location information and do not rely on any additional hardware or location/time information. However, they could cause large overhead and be less accurate compared to those approaches.

2.5.6 Centralized Approaches

Centralized approaches rely on gathering information such as statistics and visual analysis on the network connectivity graph and processing them at a central entity. It is always preferred to have the protocol decentralized and distributed. This does not require a central entity, which is a big advantage. The protocol should be localized and only rely on information collected by the nodes or their neighbors. With centralized approaches the nodes may need to access the central entity frequently. This could cause a bottleneck and delay. Also, the central entity could be a single point of failure and a tempting target for attackers to disrupt the network operation.

Most of the available protocols used decentralized and distributed approaches. Thus in our description of the available protocols the default is that the protocols are decentralized unless otherwise we specifically mention that a protocol is centralized or requires some central entity to perform some tasks.

2.6 PREVIOUSLY PROPOSED PROTOCOLS

A brief description of most of the previously proposed protocols will be provided here. Some of these protocols used more than one technique to detect wormhole attacks in ad hoc networks. We will specify the technique(s) used in each protocol. Besides the fact that a protocol will suffer from the same disadvantages related to the used technique, we will specify any other challenges related specifically with that protocol.

Hu et al. [9, 38], who introduced wormhole attacks in ad hoc networks, suggested the use of geographical or temporal packet leashes to detect wormholes. A geographical leash (location-and time-based approach) requires each node to know its own location and all nodes to have loosely time synchronized clocks. When sending a packet, the sender will include his location p_s and the time at which the packet is sent t_s. The receiving node compares these values with his location p_r and the time the packet was received t_r. The clocks of the sender and the receiver are synchronized within $\pm\Delta$, and v is the maximum node speed. A receiver node can then ensure that a sender is within a certain distance $d_{sr} \leq \|p_s - p_r\| + 2v \cdot (t_r - t_s + \Delta) + \delta$, where δ is the maximum relative error in location information, and detect discrepancies therein. The nodes need to securely exchange the information and have to authenticate the location and time information. The results will also depend on the accuracy of location estimation (δ) and the synchronization accuracy (Δ)of the clocks.

Temporal leashes (time-based approach) were also proposed by Hu et al. [9, 38]. Here, all nodes must have tightly synchronized clocks. The receiver will compare the receiving time with the sending time attached with the packet. The synchronization accuracy (Δ) of the clocks in this case have to be in the order of few microseconds. The receiver then can determine if the packet has traveled too far in too little time and detect the wormhole attack. The speed of light will be used besides the approximation for the transmission time. An alternative could be to have the source do the approximation and include an expiration time in each packet. However, the delay that could be caused by the MAC protocol may have an impact on the accuracy of the temporal leashes approach. This is besides the fact that temporal packet leashes may not detect physical layer wormholes.

Capkun et al. [3] presented a protocol (distance bounding approach) that is based on distance bounding and does not require synchronization or location information to prevent wormhole attacks. However, they depend on a secure challenge request-response and require accurate time measurements. They assumed that the network operates with central authority that controls the network membership and assigns unique identity to each node. In this case node a will send a one bit challenge to estimate the distance to node b that must respond instantaneously. Node a will detect if node b is a neighbor by using the time of flight.

Directional antennas have been used in [1] (special hardware approach) to prevent wormhole attacks. They assumed that the antennas on all nodes are aligned (which may be difficult in practice) and share a secret key with each other. A secure list of neighbors with their signal direction has to be maintained and updated at each node to detect wormholes. If a packet is received from a node that is outside the neighbor set it will be ignored. Only nodes that are in an opposite direction relative to the wormhole transceivers will accept similar nodes as neighbors. Thus, the wormhole attack's effectiveness is reduced but not fully eliminated. Not all wormhole attacks will be detected even by extending the simple directional protocol to a verified neighbor discovery protocol, that requires the co-operation of neighboring nodes. If the attacker entities are equipped with more than one transceiver at each side, and operate intelligently then the protocol will not be able to detect it.

Khalil et al. have developed two protocols to defend against wormholes: LITEWORP [31] and MOBIWORP [32]. LITEWORP (time-based and neighbor information approach) works with a static network and assumes that there is a guard node within the transmission range of any two neighboring nodes. At the beginning, each node will discover it neighbors and then broadcast its neighbors list to all of its neighbors. This will be done only once in the lifetime of each node. As a results each node will know all its direct neighbors and all the neighbors of all its direct neighbors. Also the second hop neighbor information is needed for the detection process. Each node X that is in the range of two other nodes, A and B, will act as a guard node over the link $A \leftrightarrow B$ between the two nodes. Each guard node will save information of each packet sent from B to A in a watch buffer. The information will include the packet identification and type, the packet source, the packet destination, the packet's immediate sender, and the packet's immediate receiver. The guards expect the packets to be forwarded to their ultimate destination within a time stamp. Thus LITEWORP requires overhead in terms of guard nodes and a dense network for successful operation.

MOBIWORP (time and location-based, central entity, and neighbor information approach) works with mobile networks but requires location information, a trusted central authority, and assumes the network to be loosely time synchronized. The central authority is responsible for position tracking of the mobile nodes and keeping track of adversarial behavior by a mobile node. MOBIWORP includes two types of detection: local and global. The local detection of a malicious node is done by the guard nodes in its local neighborhood, similar to LITEWORP. The global detection is done by the central authority by aggregating reports from guards at multiple locations.

In [30] a timing based defense mechanism (TrueLink) (time-based approach) against wormhole attacks is presented. Existing MAC layer acknowledgments are used to detect a wormhole. TrueLink consists of two phases: rendezvous and authentication phase. The rendezvous phase is similar to RTS-CTS-DATA-ACK exchange, here node a and node b will exchange two randomly generated numbers, called nonces. The assumption is that the IEEE 802.11 standard makes it extremely difficult for an attacker to successfully relay the exchanged frames. The authentication phase will verify that the nonces were originated from nodes a and b by making them sign and transmit the nonces. However, the protocol cannot detect physical layer wormholes.

A centralized protocol to detect wormholes in sensor networks was presented in [39] (distance bounding and centralized approach). Here, the network is reconstructed using multi-dimensional scaling and the wormhole is detected by visualizing the anomalies introduced by the attack. At first, an inaccurate distance measurement approach between the sensors that can hear each other is used to estimate the distance between every sensor pair. Then the network is reconstructed using multi-dimensional scaling to calculate a virtual position of each sensor. The estimation errors on the reconstruction are compensated using a surface smoothing mechanism. Finally, fake neighbor connections will be identified by analyzing the shape of the reconstructed network at a central entity.

Poovendran and Lazos [22] (centralized and location-based approach) presented a graph theoretic framework for modeling wormhole links and derive the necessary and sufficient conditions to detect and defend against wormhole attacks. The authors also proposed a cryptographic mechanism based on local broadcast keys to prevent wormhole attacks. The protocol requires a fraction of the nodes on the network to know their location.

Qian et al. [40] (centralized and connectivity information approach) presented a scheme to detect wormhole attacks based on statistical analysis. Here the values of routing and connectivity statistics before the attack (when the system is normal) are compared with the corresponding values after the attack. This assumes that the wormhole does not exist at the time they gather the statistics and that the statistics do not change due to other causes (e.g., mobility).

A protocol that is employing connectivity information to detect wormholes is presented in [4] (connectivity information approach). The protocol does not rely on location nor on tight synchronization, but needs topology information. The protocol looks for forbidden substructures in the connectivity graph that should not be present in a legal connectivity graph. The wormhole transceivers (M_1 and M_2) must connect a minimum number of nodes to form a forbidden substructure. At least two independent (non-neighbor) nodes at M_1's side must share three common neighbors, that are also independent, located at M_2's side. However, the protocol depends on the density of the nodes in the network and requires the network to be highly connected. Detection is not guaranteed without the availability of a specific number of independent neighbors. In fact, even in a network where the nodes are densely deployed and highly connected if the transmission range of the wormhole transceivers is short then the wormhole will not be detected. Such a wormhole will still have the same impact of a wormhole with large transceivers' range. We compare our results with the results presented in [4] in Section 3.9.4.

The node degree is used to detect wormholes in [5] (connectivity information approach). The assumption here is that the wormhole will increase the number of one-hop neighbors of a node and if this number is greater than some threshold (could be the average node degree) then the node will need to check for a wormhole. If however the wormhole connects a single node with another node that is far away, the node degree only changes by one and the wormhole will not be detected. Another possibility could be to place the wormhole between nodes that have a node degree less than the average, thus will not be detected. But the damage to the network is comparable to a wormhole connecting a group of nodes. Also, the protocol suggests an approximate removal process for a set of suspicious links that may completely isolate some nodes from the network.

In [6], an approach similar to [5] was presented. Again the assumption made is that the wormhole will significantly increase the number of one-hop neighbors. Also nodes are assumed to be uniformly and densely deployed with no links changed or added. Each node will count the number of nodes that are two-hops away and the idea is that this number grows under a wormhole attack.

Su and Boppana [27] (distance bounding approach) proposed a distributed technique to detect in-band wormhole attacks in mobile ad hoc networks. The protocol is based on the propagation speeds of requests and statistical profiling. They do not require the clocks to be synchronized network-wide and no additional control packets are needed. The protocol is supposed to be complementary to the existing source routing protocols.

In [41] (time-based and distance bounding approach) researchers proposed a mechanism to detect and avoid wormhole attacks in the OLSR protocol. The idea is first to detect network links with high probability to be involved in a wormhole attack. This is done based on the assumption that wormhole attacks will cause longer packet latency compared to a single hop normal wireless latency. This could be true if the wormhole is in-band, using nodes in the network to tunnel packets. Afterwards, suspicious links will be challenged by exchanging encrypted probing packets between the two ends of the link. If the two nodes are actually connected through a wormhole then the response is expected to be slower than legal links. A similar approach that can detect tunneling of packets in dynamic source routing (DSR) is presented in [42].

In [43] (neighbor information and centralized approach) researchers proposed two detection mechanisms for wormholes in wireless sensor networks. The first mechanism detects the increase in the number of the neighbors of the nodes. This is assumed to be due to the new links created by the wormhole in the network. The second mechanism detects the decrease of the lengths of the shortest paths between all pair of nodes. This is assumed to be caused by the short links created by the wormhole. All the nodes in the network will send their neighbor list to the base station. The base station is responsible to reconstruct the network graph from the received neighborhood information and detect the wormhole.

In [11] (time-based approach) researchers proposed a technique that is based only on synchronized clocks to detect man-in-the-middle attacks. They used secure clock synchronization in fixed wireless networks. It was also shown that no timing based solution can detect an optimal attacker. Their definition for the optimal attacker is similar to the physical wormhole defined earlier in this chapter. The optimal attacker is equipped with double full-duplex radio and can inject a constant delay.

[44] (connectivity and neighborhood information), the protocol used to detect worm-

hole attacks employs local neighborhood information. The network topology is assumed to be static and the links are assumed to be bidirectional. However, they assumed that the wormhole must change the topology structure of the network and they computed a so-called edge-clustering coefficient. The assumption is that in a dense network every two neighbors must have a common neighbor. A wormhole node is detected by one of its neighbors if that neighbor cannot reach one of the wormhole neighbors without using that node. However, it is very possible to come up with many scenarios with wormholes that will not satisfy any of the necessary conditions with this approach to detect the wormhole. This will only successfully detect open wormholes or closed wormholes that only connect one single node with another single node. If the wormhole connects a group of nodes (≥ 2) with another group of nodes, which is the most common form of wormhole, then the protocol will not detect the wormhole.

A framework to detect wormhole attacks in wireless ad hoc networks was proposed in [45] (time-based approach). The detection consists of two phases. The first phase is supposed to be inexpensive, referred to as "suspicion", and must detect the wormhole. Two techniques are used in this phase to detect the wormhole – RTT (round trip time) and topology information. The assumption is that if a wormhole exists, then abnormal RTT (higher than average) between nodes connected with wormhole must be observed (only true in in-band wormholes). Moreover, neighbors in dense networks must always have common neighbors. Only if something is detected with phase one then phase two (confirmation) will be used. In phase two the attacker will be challenged by either using frequency hopping or the TrueLink technique [30]. Obviously, none of the techniques used in this framework will guarantee the detection of wormholes for all cases.

In [46] (time-based approach) authors proposed a wormhole detection mechanism that relies on delay measurements. The idea is to obtain the delay and the hop count information of some disjoint paths between the sender and the receiver. Then, they use the delay/hop value to detect the existence of a wormhole in any of the disjoint paths. The assumption is that the delay in the wormhole corrupted path will be unreasonably high. Thus, a path with a distinguishably high delay/hop value is more likely to have a wormhole attack.

In [47] (time-based approach) authors proposed a transmission time based mechanism

(TTM) to detect wormhole attacks. The protocol requires the computation of the transmission time between every two successive nodes along the established path during route setup procedure. The assumption is that the transmission time between two fake neighbors created by wormhole is considerably higher than that between two real neighbors which are within radio range of each other.

In [48] (location-based approach) an end-to-end wormhole attack detection is proposed. Based on geographic information exchanged between the source and the destination, the source node estimates the minimum hop count to the destination. The source compares the hop count value, for each used route, received from the reply packet with its estimated value. If the received value is less than the estimation, the corresponding route is marked as if a wormhole is detected. Afterwards, the source launches wormhole TRACING in which the two end points of the wormhole will be identified in a small area given that there are multi-paths that exist between the source and destination.

A secure neighbor verification protocol for wireless sensor networks is proposed in [49] (distance bounding and neighbor information approach). Their protocol is distributed and relied on estimated distances between nodes. They require each node to be equipped with a microsecond precision clock and two network interfaces: a radio-frequency and a sound interface. At first, nodes will discover their neighbors insecurely, that is, without any guarantee that all neighbors are real. Then each node will estimate the distance to all its neighbors using an ultrasound-based ranging protocol. After that, each node exchanges its neighbors information including the estimated distance, with its neighbors, in a secure way. Each node will then create a table the includes distances between its neighbors. Finally, the created neighboring table will be examined by a number of security tests and each link is marked as either verified or unverifiable.

In [50] researchers proposed a protocol for neighbor discovery in ad hoc networks using directional antennas. However, the protocol did not discuss or address any attacks including the wormhole attack. Also [51] merged a MAC protocol with neighbor discovery protocol for ad hoc networks using directional antennas. The researchers focused on the throughput of the protocol and did not address the wormhole attack.

3.0 THE DEWORM PROTOCOL

3.1 INTRODUCTION

This chapter will present and discuss the proposed protocol to detect one of the most devastating attacks in wireless ad hoc networks. The previous Chapter showed that most of the previously proposed techniques to detect wormhole attacks require precise and accurate information about the location of nodes, the time of packet transmission and synchronization between nodes, or the use of special hardware (e.g., directional antennas). Protocols that depend on location information require the nodes to be equipped with GPS [9], which may not provide the required location accuracy, especially in indoor and urban areas. Defense mechanisms that rely on time measurement and synchronized clocks cannot detect physical layer wormholes. Protocols that use special hardware such as directional antennas [1], special RF [3], or ultrasound [2] add expense, complexity, and need for special customization.

In this dissertation, we propose "DeWorm", a novel, yet simple protocol to effectively detect wormhole attacks. We employ routing discrepancies between neighboring nodes along a path from a source to the destination to detect wormhole attacks. The protocol is simple and localized, can be applied on demand (when the existence or lack thereof of a wormhole needs to be verified) and needs no special hardware, localization, or synchronization. Thus DeWorm can detect physical layer wormholes. We have tested the protocol and evaluated it with grid-like and randomly distributed networks with various connectivity models (unit disk graph - UDG, Quasi-UDG, both with symmetric and asymmetric links). Our simulations show that DeWorm has nearly 99% detection probability and very few false positives. The cost of the protocol is the addition of a little overhead and that only when deployed.

The example in Figure 3.1 summarizes the idea of the DeWorm protocol. Here the source

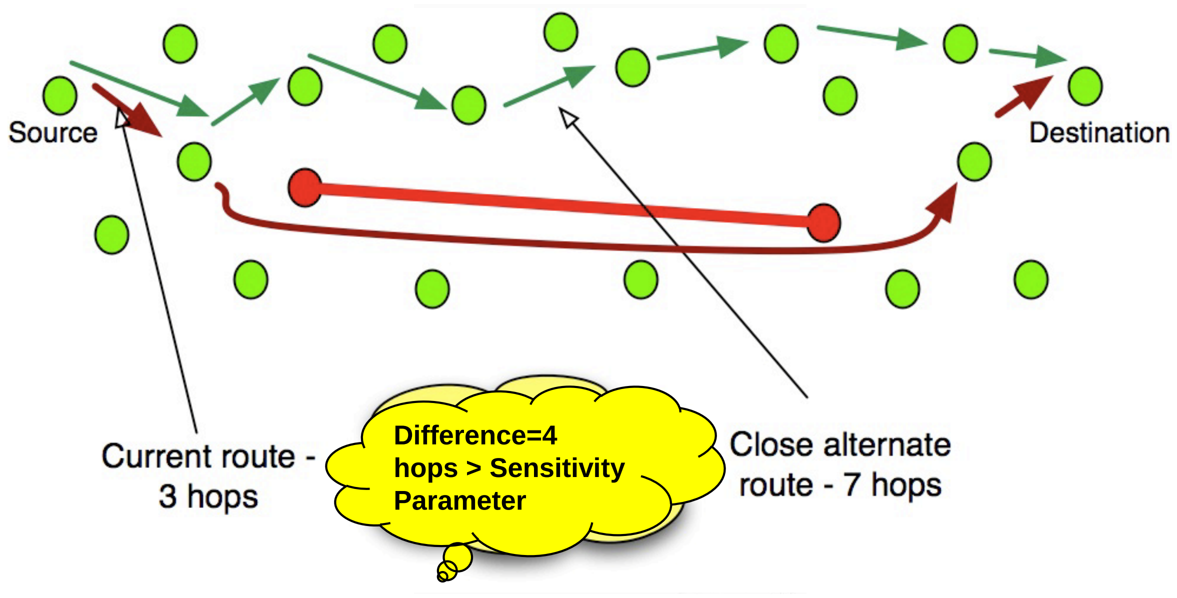
Source
Destination
Current route - 3 hops
Difference=4 hops > Sensitivity Parameter
Close alternate route - 7 hops

Figure 3.1: The Idea of DeWorm

has a short route (the route in red color) to the destination that goes through the wormhole. The goal of DeWorm is to find alternative route that does not go through the wormhole (the route in green color). The source will check the difference between the two routes and if this difference is greater than some value that we will refer to as the sensitivity parameter then a wormhole is detected. In this case the difference is 7-3 = 4, we will use a sensitivity parameter as 2 or 3.

The difference is some how related to the length of the wormhole in number of hops. This explains why 2 or 3 is a good choice for the sensitivity parameter. Note that by definition wormholes are longer than 1 hop (at least 2 hops). In fact the attacker wants to make the wormhole as long as possible because the longer the wormhole the better the attack since it will attract more traffic and will have larger impact.

The approach DeWorm is using sounds very simple in this example but HOW can we guarantee to find alternative route that will not go through wormhole so that it will have a significant difference in length compared to the route that goes through the wormhole? This question and the detailed description of DeWorm operation will be clearly explained in this chapter. The rest of the chapter is organized as follows. In the coming sections a a brief description of DeWorm is presented this will be followed by a detailed description and a step-by-step example for DeWorm detection. In this second part of the chapter we will discuss different aspects of DeWorm and provide our evaluations and simulation results.

3.2 BRIEF DESCRIPTION OF DEWORM

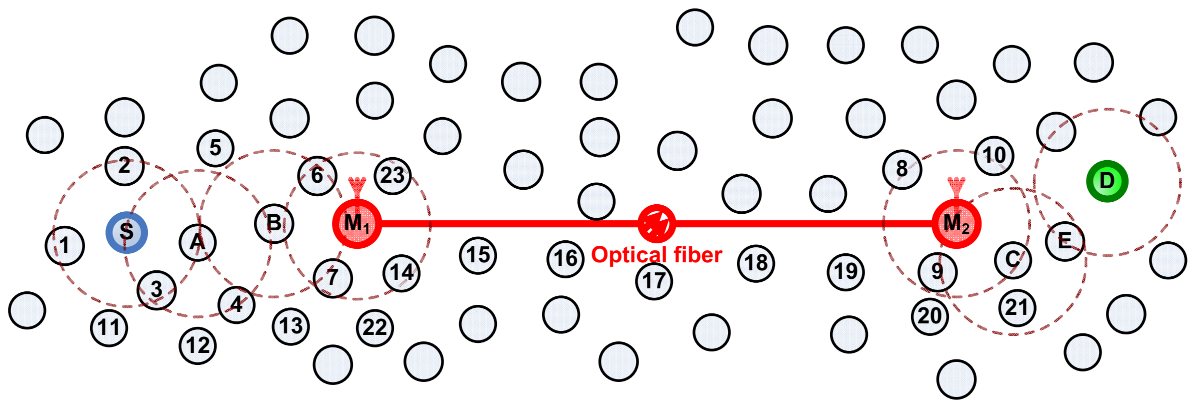

Figure 3.2: Wormhole detection example

The sender node S in Figure 3.2 will initially have a route to the destination node D and wishes to test whether this route includes a wormhole or not. We assume this is a physical layer wormhole (e.g., a high-speed fiber link) with nodes M_1 and M_2 acting as relays and otherwise invisible to honest nodes (they do not advertise their IDs or MAC adresses). Detecting such wormholes is considered to be extremely difficult [31]. The sender S will start by discovering his one-hop neighbors. Based on the received replies, he will create a list of his one-hop neighbors that excludes the next hop along the route. The sender will check the routes (we call these the test routes) that are used by these one-hop neighbors to the *second hop* along the route to the destination (throughout this paper we will refer to this node as the **target node**). Node S compares the length of a *selected* route with the one he has to the target node. The selected route is chosen from the routes reported from the neighbors. Later in Section 4.3.1 we will discuss methods that can be be used to determine the selected route for comparison. If the difference between the numbers of hops of the two routes is greater than a certain value called the "sensitivity parameter", the sender will assume that a wormhole exists. If not, this process is repeated by each node that lies on the route (such nodes also exclude the previous hop from the list). The idea is that when a node that is close to M_1 is reached, its next hop neighbor along the route will be on the other side of the wormhole link (near M_2). If at least one of the "perceived" one-hop neighbors is located within the transmission range of the node, (i.e., it is not on the other side of the wormhole), the route from this neighbor to the target node can be rendered very different (typically long) and thus the wormhole will be detected.

The general idea for the detection algorithm to work is as follows. We need to ensure that not all the picked neighbors are near M_2 nor will reach the target node using the wormhole link. If not, the selected route to the target node will be comparable in length to the route that the sender has, and the wormhole attack will not be detected. Yet another problem is that the picked neighbors may also have a route to the target node through another node that uses the wormhole link. Again, in this case the length of the selected route will not be significantly different to detect the wormhole. To prevent this situation, the sender provides each neighbor with a list of nodes to avoid in their route to destination. This list will include all nodes that are within the range of the wormhole's transceiver at the far side of the network. Consequently, legitimate one-hop neighbors will avoid routes through the wormhole link. Details follow.

3.3 DETAILED DESCRIPTION OF DEWORM

Consider a communicating source-destination node pair $(S, D) \in W$, with route $\mathcal{P}_{S,D}$. If node S wishes to detect the existence of a wormhole, a naive approach would be to delete the nodes in the current route $\mathcal{P}_{S,D}$ from consideration. Next, S would discover a new route to D and if the length of the new route differs significantly compared to the length of $\mathcal{P}_{S,D}$ (i.e., greater than a threshold), it concludes a wormhole exists. This approach unfortunately will not work because it is difficult to ensure that the alternate route does not traverse through the wormhole as well (in which case the two routes will have similar lengths). Ensuring that the wormhole is avoided is difficult because S will have no idea as to the location of the wormhole along the route $\mathcal{P}_{S,D}$. Furthermore, as illustrated in Fig.3.2 the wormhole will typically connect several nodes (i.e., B_{M_1} and B_{M_2}) and it is likely that the alternate end-to-end route between S and D will also pass through the wormhole. In order to avoid this problem DeWorm works through the nodes in $\mathcal{P}_{S,D}$ in a sliding fashion checking the length of alternate routes between nodes that are a short distance apart (2 hops typically) and employs a forbidden list approach to avoid neighbor nodes possibly in range of the wormhole (i.e., B_{M_2}).

A flow chart of the DeWorm protocol is shown in Fig.3.3. We discuss the steps in the flowchart below. Consider a source node S that wants to communicate with destination node D and wishes to test for a wormhole. Let $u, v, x \in \mathcal{P}_{S,D}$ – they are nodes on the path from S to D that was obtained by using some standard routing protocol. Let the wormhole $M_1 \leftrightarrow M_2$ connect nodes u and v where $u \in B_{M_1}$ and $v \in B_{M_2}$. Let x be the next hop from v on the route from S to D. Note that u and v are typically separated by several hops, but now will believe that they are neighbors.

Step(1): The "Sender" node S will set the target node T to be the node two hops away along the path, i.e., $T = \mathcal{P}^2_{S,D}$ initially.

Step(2): S will discover all its one-hop neighbors B_S by broadcasting a "hello" message. The nodes in B_S will hear the hello message and will reply to S.

Step(3): S will create a list of the nodes in B_S and marks node $\mathcal{P}^1_{S,D}$. Note that node $\mathcal{P}^1_{S,D} \in \mathcal{P}_{S,D}$ and is known during route discovery to the destination D.

Step(4): S will broadcast the list (B_S, T) and ask every node $q \in \{B_S - \mathcal{P}_{S,D}\}$ to find a route to target node T, such that the route does not include any other node in B_S (will be referred to as forbidden list). That is $\forall q, z \in B_S$ where $q \neq z$, DeWorm ensures that $z \notin \mathcal{P}_{q,T}$. Each node $q \in \{B_S - \mathcal{P}_{S,D}\}$ will run the network routing algorithm and reply to S with l_{qT}, the length (in number of hops, or the cost) of its route to T. If l_{qT} does not exist due to the connectivity of the network topology then q will inform S and S discards q from $\{B_S - \mathcal{P}_{S,D}\}$.

Step(5): The sender will pick a "selected route" and determine its length. For example,

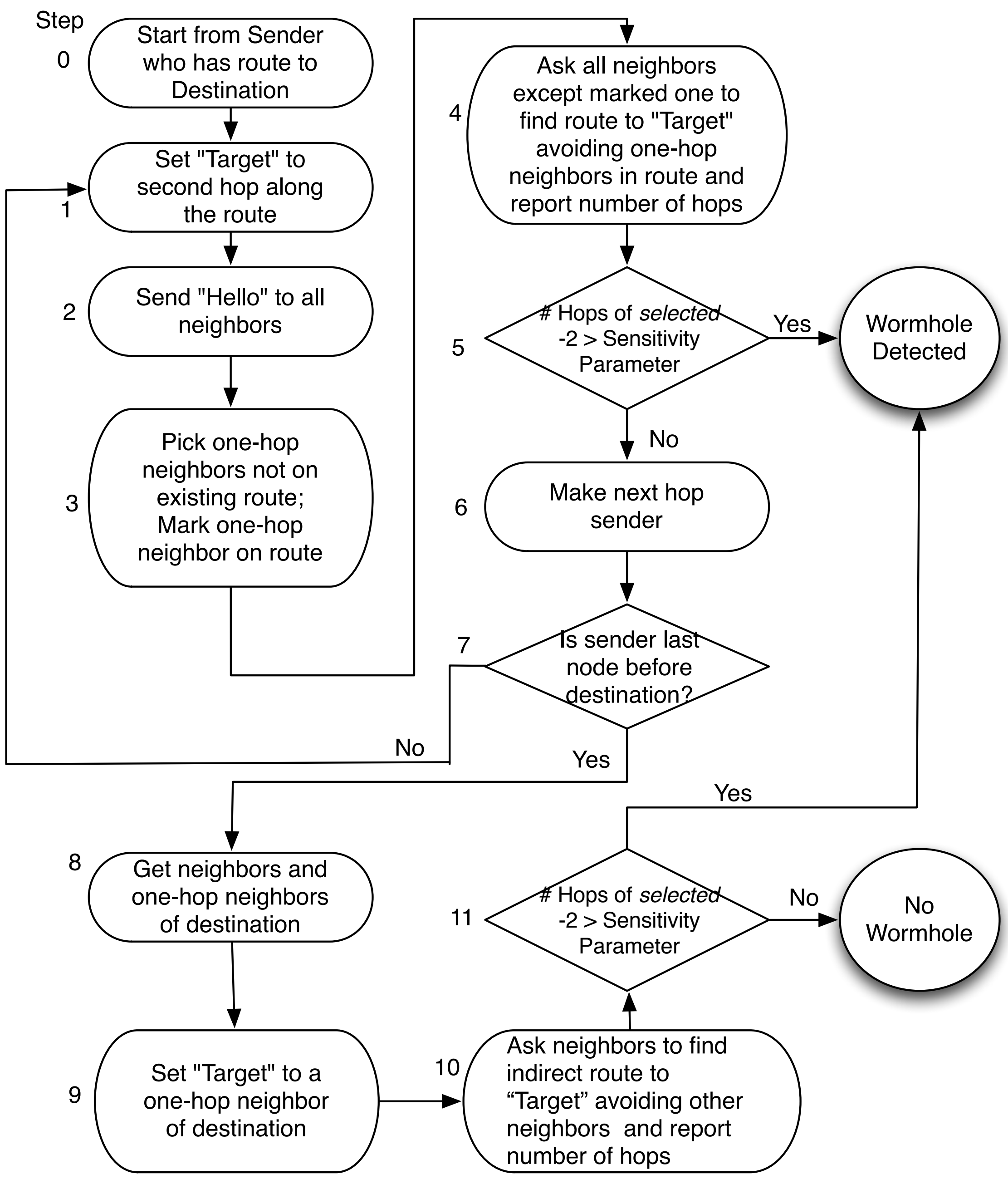
Step
0
Start from Sender who has route to Destination
1
Set "Target" to second hop along the route
2
Send "Hello" to all neighbors
3
Pick one-hop neighbors not on existing route; Mark one-hop neighbor on route
4
Ask all neighbors except marked one to find route to "Target" avoiding one-hop neighbors in route and report number of hops
5
# Hops of *selected* -2 > Sensitivity Parameter
Yes
Wormhole Detected
No
6
Make next hop sender
7
Is sender last node before destination?
No
Yes
8
Get neighbors and one-hop neighbors of destination
9
Set "Target" to a one-hop neighbor of destination
10
Ask neighbors to find indirect route to "Target" avoiding other neighbors and report number of hops
11
# Hops of *selected* -2 > Sensitivity Parameter
Yes
No
No Wormhole

Figure 3.3: Flow chart for the DeWorm protocol

$L = Max\{l_{qT}\}\,\forall q \in \{B_S - \mathcal{P}_{S,D}\}$, the length of the longest route can be used (other options for L are discussed in Section 4.3.1). The sender tests for the existence of a wormhole by comparing the length L of the "selected route" to T with the direct route. Specifically, if $L - 2 > \eta$ then the sender will assume that a wormhole is detected. Note, that $L - 2$ is used because T is 2 hops away from S and η is a tunable sensitivity parameter (see Section 4.3.1). If a wormhole is detected then DeWorm stops.

Step(6): If no wormhole is detected then one increments the procedure to the next hop along the route (e.g., node $\mathcal{P}^1_{S,D}$ will become the new "sender" S and $\mathcal{P}^3_{S,D}$ becomes the new target T).

Step(7): If the new sender is *not* the last node on the route before the destination D (i.e., $S \neq \mathcal{P}^{l_{SD}-1}_{S,D}$), then steps numbered 1 to 6 will be repeated by the new "sender".

Comment: DeWorm will detect the wormhole when a node that is within M_1's range becomes the "sender". In this case node $u \in B_{M_1}$ will perform steps 1 to 6. In Step 2, u determines its neighbor list $B_u = \left\{\hat{B_u} \cup B_{M_2}\right\}$ which includes not only the true one-hop neighbors $\hat{B_u}$ that are within node u's range but also all the nodes that are within M_2's range. Node u will ask all nodes $q \in \{B_u - \mathcal{P}_{S,D}\}$ to find routes to the target node $T = x$ that *appears* to be two hops away – it is actually one-hop from v – such that the routes exclude nodes in B_u. When a node $q \in \hat{B_u}$, i.e., a true one-hop neighbor, tries to find a route to node $T = x$, it also avoids routes containing the nodes in B_{M_2}. Thus the reported $l_{qT} \geq (2 + length\ of\ wormhole)$. If the wormhole spans a length greater than η and $L = Max\{l_{qT}\}\,\forall q \in \{B_S - \mathcal{P}_{S,D}\}$ then $L - 2 > \eta$ and the wormhole is detected.

Special Case: Steps 8-12 in the lower part of the DeWorm flowchart Fig.3.3 are activated only if a wormhole is not detected and the "sender" becomes the last node before the destination $S = \mathcal{P}^{l_{SD}-1}_{S,D}$. This will occur when $\mathcal{P}_{S,D}$ does not pass through a wormhole or if the destination node D is next to the wormhole (i.e., $D \in B_{M_2}$). In either event the DeWorm protocol follows the procedure below.

Step(8): Node $S = \mathcal{P}^{l_{SD}-1}_{S,D}$ will discover all its one-hop neighbors B_S by broadcasting a "hello" message and will ask node D to provide its one-hop neighbor's list B_D.

Step(9): Node S will set as the "Target" a node that is D's neighbor. That is $T \in B_D$.

Step(10): Node S will broadcast the list $(B_S \cup B_D, T)$ and ask every node $q \in B_S$ (except

D) to find an indirect path to T (has to pass through at least one other node before reaching T), which does not include any other node in $\{B_S \cup B_D\}$. That is $\forall q, z \in B_S$ where $q \neq z$ DeWorm ensures that $z \notin \mathcal{P}_{q,T}$ and $l_{qT} > 1$. Each node $q \in B_S$ will run the network routing algorithm and reply to S with l_{qT}.

Step(11): The sender will pick a "selected route" and determine its length. For example, $L = Max\,\{l_{qT}\}\, \forall q \in \{B_S - \mathcal{P}_{S,D}\}$. Again, if $L - 2 > \eta$ then the sender will assume that a wormhole is detected.

Comment: If there is wormhole connecting node $u = \mathcal{P}_{S,D}^{l_{SD}-1}$ with node D then node $u \in B_{M_1}$ will have a neighbor list $B_u = \left\{\hat{B_u} \cup B_{M_2}\right\}$ which includes not only the true one-hop neighbors $\hat{B_u}$ within node u's range but also all the nodes within M_2's range. Similarly, node $D \in B_{M_2}$ will have its neighbor list $B_D = \left\{\hat{B_D} \cup B_{M_1}\right\}$ with both the true one-hop neighbors $\hat{B_D}$ and the nodes that are within M_1's range. Node u will ask all nodes $q \in B_u$ (except D) to find indirect routes to the target node $T \in \{B_D\}$ that will avoid nodes in $\{B_D \cup B_u\}$. Note that both B_{M_1} and B_{M_2} are $\subset \{B_D \cup B_u\}$. Depending on the location of T we will have two cases and in both, node u will detect the wormhole. In the first case node u may pick node $T \in \hat{B_D}$ but $\notin B_{M_1}$. The one-hop neighbors of u that are in $\hat{B_u}$ will have long indirect routes to T while those in B_{M_2} will not. In the second case node u may pick node $T \in B_{M_1}$ but $\notin \hat{B_D}$. Then, the one-hop neighbors of u that are in B_{M_2} will have long indirect routes to T. Consequently, in both cases there will be a long route from a neighbor to T and the wormhole will be successfully detected.

3.4 DEWORM BY EXAMPLE

Next we describe the steps of our protocol in more detail. A flow chart of the protocol is shown in Figure 3.3. The example we use to clarify how DeWorm works will follow the network shown in Figure 3.2.

Step(1): Let us suppose that node S wants to communicate with node D and the shortest path provided by some standard routing protocol is (S-A-B-C-E-D). Note that there are five hops to the destination. Obviously this route passes through the wormhole and nodes

B and C are connected through the wormhole transceivers M_1 and M_2 without being aware of this fact. Our goal is to show that using DeWorm, node S will be able to detect the wormhole.

Step(2): The sender will discover all his one-hop neighbors by broadcasting a "hello" message.

Step(3): The one-hop neighbors of the sender will hear the hello message and will reply to the sender. In this example, node S will receive replies from nodes A, *1*, *2*, and *3*. Upon receiving the responses, S will add the nodes to his one-hop neighbors list.

Step(4): The sender will create a list of one-hop neighbors and marks node A. Note that node A is the next hop on the already determined route to the destination D. The sender will send this neighbor list including the marked node to the rest of the neighbors (in this case, nodes *1*, *2*, and *3*). The sender will ask these nodes to find a route to the target node, in this case node B, which does not go through any node from this list. The one-hop neighbor nodes will reply to the sender with the length (in number of hops, or the cost) of the route to B. Note that nodes *1*, *2*, and *3* are required to find a route to B that does not go through nodes S, A, *1*, *2* and *3*. In our example, nodes *1*, *2*, and *3* will find routes to B as: (*1-11-12-4-B*), (*2-5-B*), and (*3-4-B*) and they will inform the sender S that the lengths of the routes to B are 4, 2, and 2 hops, respectively. The sender will pick the longest route as the "selected" route (or use another method – see Section 3.5.1) with 4 hops here, and compares it with 2 hops, the distance to the target node. If a one-hop neighbor does not find a route to the target node that does not include nodes from the list, it will inform the sender. The sender disregards such nodes.

Step(5): If the number of hops of the selected route minus 2 hops is greater than the sensitivity parameter (chosen as 2 in this example) then the sender will assume that a wormhole is detected. In this example the selected route minus 2 will be 4 - 2 = 2 which is not greater than the sensitivity parameter. Thus, no wormhole is detected.

Step(6): The next hop – node A – will become the new "sender" (there is now a new target as well – node C).

Step(7): Steps numbered 2 to 6 will be repeated by the new sender until either a wormhole is detected or the destination node is reached (i.e., the sender node becomes the last node on the route before the destination D).

In our example, node A will pick nodes *3* and *4*. The routes from nodes *3* and *4* to C (excluding nodes S, *3*, *4*, and B) will be (*3-12-13-7-*C) and (*4*, *13*, *7*, C), respectively. The selected route minus 2 will be 4 - 2 = 2, which is again not greater than 2. Thus the wormhole is still not detected and the new sender will be the next hop, node B.

Node B is within the transmission range of M_1, as shown in Figure 3.2. Node B will send a "Hello" message to discover its neighbors. M_1 will receive the message and send it using the fiber link to node M_2, which will then broadcast it to its neighbors. As a result node B will have nodes A, *4*, *6*, *7*, *8*, *9*, *10*, and C in its one-hop neighbors list. Note that the replies from nodes C, *8*, *9*, and *10* are transmitted by M_1. Nodes *4*, *6*, *7*, *8*, *9*, and *10* will all try to find routes to node E that do not pass through the one-hop neighbors' list of node B. Since all the nodes that are within the transmission range on the other side of the wormhole M_2 (nodes C, *8*, *9*, and *10*) are in the forbidden list, any route from nodes *4*, *6*, or *7* will not pass through the wormhole and will be long enough to detect the wormhole. The selected route will be from node *4* that is (*4*, *13*, *22*, *14*, *15*, *16*, *17*, *18*, *19*, *20*, *21*, E) which has 11 hops. Thus in this case we have 11 - 2 = 9 which is greater than 2 and consequently node B will inform node S that a wormhole has been detected.

Special Case: When the destination node is within M_2's transmission range, the "sender" who has the destination node as the target node will not be a one-hop neighbor of M_1. Such a sender will thus not have the nodes in M_2's transmission range, those nodes on the other side of the wormhole, in his one-hop neighboring list. These nodes will not be in the forbidden list that he will send to his one-hop neighbors and so they will be part of any route (likely through the wormhole) to the target. Consequently, the sender will not detect the wormhole. In the example in Figure 3.2, let us suppose that the destination is node C and that node A was not able to detect the wormhole. We need the last node on the route, right before the destination node, (this will be node B in the example) to detect the wormhole. Node B will have nodes A, *4*, *6*, *7*, *8*, *9*, *10*, and C in its one-hop neighbors list. Note that this list includes all nodes that are within M_2's range (nodes *8*, *9*, *10*, and C). Node B (the last node on the route) will ask node C (the destination node in this case) to provide its one-hop neighbors' list, which will contain *9*, *21*, E, B, *6*, *7*, *14*, and *23*. Note that, this list includes all nodes that are within M_1's range (nodes B, *6*, *7*, *14*, and *23*). Node B will pick any node from node's C one-hop list excluding nodes that are in its own one-hop list and will make it the target node. Thus, node B could pick any of nodes *21*, E, *14*, and *23* as the target node. Next, node B will ask its one-hop neighbors (except node C) to find an indirect path to the target node (has to pass through at least one node before reaching the target node) excluding the common one-hop neighbors of nodes B and C. This forbidden list will have nodes A, *4*, *6*, *7*, *8*, *9*, *10*, C, *21*, E, B, *14*, and *23* (this list includes nodes that are within M_1's or M_2's transmission range).

Depending on the location of the target node we will have two cases and in both, node B will detect the wormhole. In the first case node B may pick a target node that is on M_2's side of the network (node *21* or E). In this case the one-hop neighbors of node B that are located on M_1's side (nodes *4*, *6*, *7*) will have long indirect routes to node *21* or E. We recall that nodes A, *4*, *6*, *7*, *8*, *9*, *10*, C, *21*, E, B, *14*, and *23* are included in the forbidden list and cannot be used to reach *21* or E). In the second case node B may pick a target node that is on M_1's side of the network (node *14* or *23*). In this case the perceived one-hop neighbors of node B that are located on M_2's side (nodes *8*, *9*, *10*) will have long indirect routes to node *14* or *23*. Again, we note that nodes A, *4*, *6*, *7*, *8*, *9*, *10*, C, *21*, E, B, *14*, and *23* are included in the forbidden list and cannot be used to reach *14* or *23*. Thus in both cases there will be a long route that is compared to the single hop and the wormhole will be successfully detected. This special check needs to be performed by the last node along the route before the destination only if the wormhole is not detected previously.

In our description and detailed example, we have used a unit disk graph (UDG) as our connectivity model and assumed the links are symmetric. With a Quasi-UDG [52] connectivity model or in the presence of asymmetric links, DeWorm's behavior will be the same. The only difference is that the number of neighbors will be changed. In the simulation results presented in Section 3.9.3 we test DeWorm with the Quasi-UDG model and also in the presence of asymmetric links. The results show that the performance of DeWorm is not affected.

We assume that messages that are exchanged between the sender and its neighbor nodes are authenticated, and nodes M_1 and M_2 do not have shared keys with honest nodes. This prevents fabrication attacks by M_1 and M_2. Sharing of keys between nodes and related security attacks are beyond the scope of this dissertation. Please note that a wormhole attack can succeed even if nodes M_1 and M_2 do not share keys with other nodes in the network.

3.5 ANALYSIS OF DEWORM

In this section we consider different aspects of DeWorm and address potential limitations. We will discuss the different methods to pick the route selected for comparison and discuss the sensitivity parameter. Then we will illustrate the connectivity requirements and show that DeWorm does not require network with special topology or high node degree.

3.5.1 Route Selected for Comparison and Sensitivity Parameter

DeWorm, essentially has two parameters to be selected, the sensitivity parameter η and the method for determining L from the set of routes found to the target node (i.e., the "selected route" mentioned previously). Wormholes are at least longer than the transmission range of a node (otherwise their impact is minimal). With $\eta = 1$ even short wormholes can be detected. However, the number of false positives will increase. Using $\eta = 5$ reduces false positives but short wormholes may escape detection. Thus, $\eta = 2$ or 3 provides the best tradeoff between the detection rate and false positives.

There are many possibilities for the determining L from the set of routes found to the target node T for comparison with η to detect the wormhole. In our simulations we have tested several route selection methods to determine L (see section 3.5.1). In the previous section, we used the length of the longest route as the value of L to illustrate DeWorm. Another option is to use the average length of the computed routes to the target as L. The former increases the false positives while the latter reduces the detection rate (see Table 3.1). While any of these choices is unlikely to significantly impact the wormhole detection performance, we may expect differences in the number of false positives. The reason is that routes from some neighbors may be longer than those from other neighbors by a value greater than the sensitivity parameter.

Ultimately, we used a method that provides the best detection and a reasonable number of false positives. The sender creates a list from the replies containing route lengths to the target node from its neighbors and sorts them according to their lengths from longest to shortest (excluding replies from neighbors that do not have routes to the target node). The sender picks L as the length of a route that is smaller than the longest route by *not more than* η if it exists. Otherwise L is picked as the length of the longest route. We find that the shortest such route – e.g., if the longest route has 10 hops, the sensitivity parameter is 2, a route with 8 hops if it exists – is the best option. But a route with 9 hops is better than the route with 10 hops if one with 8 hops is not in the list. The reason why we do not pick the longest route when such shorter routes are available is to avoid cases when the longest routes are actually outliers. By eliminating the longest of the long routes, we reduce the number of false positives. Using η to decide whether the length of a shorter route is sufficient, ensures that we do not miss those cases where the longest route is the only long available route from a node located at M_1's side that can be used to detect the wormhole. This method is labeled "Sensitivity" in Table 3.1.

3.5.2 Connectivity Requirements

While neighbors of nodes that lie on the route need to find routes to target nodes, it is not necessary for such routes to be node-disjoint. In the case of a very low density network, where we may have a single route from the source to the destination, the node along the route that is close to M_1 should be able to pick at least two neighbors that are on opposite sides of the wormhole, that have different routes to the target node, and thus the wormhole can be detected. Our protocol does not require a very highly connected network, as compared to [4], to detect wormholes. In Section 3.9.3.4 we will show that DeWorm can still work with networks that have an average node degree of around 2, although its effectiveness improves with the node degree.

In [6, 5] researchers the nodes have to be uniformly and densely deployed so that the wormhole can be detected. DeWorm does not require any special topology or uniform distributions. The node can have different low or high node degree and the nodes can be randomly distributed. All of these cases will be tested and proven in the results section.

If a *critical node* (if this node is removed, the network will be partitioned) exists in the route from S to D then DeWorm will work only if M_1-M_2 is closer to S than the critical node by at least one hop. Otherwise, the neighbors of a node that is one hop from the critical node along the route will not be able to find any route to the target node that avoids the critical node. If critical nodes are few and don't change in the network, one possible solution to this issue is to use a protocol that can identify critical nodes [53, 54] a priori that will cooperate with DeWorm. A critical node and the node just before it along the route will then be exempt from using DeWorm.

3.5.3 Low Node Degree Modification

To maintain very high detection rates in sparse networks with extremely low node degree, we suggest the following modification. The sender will inform his neighbors of not only the target node, but also with the complete route to the destination. If a neighbor cannot find a route to the target node, then it will try to find a route to the next hop that comes after the target node along the route to the destination and so on. It will try all the nodes on the route to the destination including the destination. When the neighbor finds a route it informs the sender about the length of its route and it will also specify the target node of this route. This will also help to resolve the situation when the wormhole is connecting two separate (partitioned) networks. In this situation none of the neighboring nodes will succeed to find any route to the target node or to any of the nodes along the route to the destination. If this is the case then S can confidently assume that there is a wormhole that is connecting two separate networks.

We have simulated the case where the destination is always chosen as the target node and found that DeWorm performs equally well there. The route discovery from a neighbor to the destination may however take more time than discovery of routes to a closer target node.

3.6 OTHER ISSUES

In this section we will discuss other issues related to the DeWorm protocol. It will be shown that practically mobility will not have any impact on the detection performance of DeWorm. We will also suggest a simple technique to improve the stability of DeWorm with networks with high mobility. Moreover, we will present simple example of how DeWorm can be integrated with some mobile ad hoc routing protocols.

3.6.1 DeWorm and Mobility

In this section we will start by estimating an approximation for the time delay with DeWorm to detect a wormhole. We will start by estimating the time required by each sender along the route to the destination to check for the wormhole. Each sender needs to broadcast a message and in [55] the duration required to transmit a packet successfully was computed. In [55] they used IEEE 802.11 with RTS/CTS and channel rate of 11Mbps. Their results show that even for large packets of size 1000 Bytes, the delay will be less than 20 μS. Neighbors need to find routes to the target node, which is on average 2 to 3 hops away. In fact all the neighbors can simultaneously start finding their routes once they receive the broadcast request from the sender. In [56] it was shown that route acquisition latency depends on the length of the route. In this case the routes are all expected to be fairly short (2 or 3 hops) except for neighbors of the node along the original route that is within the range of M_1. Thus the average delay in this case can be expected to be small. Consequently, we believe that wormhole detection by DeWorm is efficient and reasonably quick.

Results in section 3.9.3 will show that DeWorm is not sensitive to changes in the connectivity model (i.e., with quasi-UDG and asymmetric links). DeWorm was not tested for mobile systems. However mobility will only have impact on the operation of DeWorm if the topology changes rapidly resulting in new neighbors arriving around M_2. This means that between the time that the "sender" node discovers its neighbors and just before the neighbors find their routes to the target node a new node moves and become within the range of M_2.

Let us assume that the neighbors will find routes to the target nodes using some flood routing technique. The routes are expected to be 2 to 3 hops, thus on average the process of checking will require the time of sending 5 packets, which will roughly take 100 to 150 μS. Given that a node that is located at the border of the transmission range of M_2 will be required to move at least 5 to 10 meters to be considered within range. Thus practically it is not possible for the neighbors list of the “sender” to be changed in 150 μS (average node speeds in ad hoc network are 2 or 5 m/s).

Since mobility may change node locations and make routes little longer, then increasing the sensitivity parameter will help DeWorm be more adaptive to handle mobility and reduce the false positives. Another possible solution for the mobility problem is to use a technique similar to the one presented in section 3.10.1 to prevent the smart wormhole attack. The idea is to make the “sender” recheck his neighbors after he gets the routes from his neighbors to the target node and ensure that the list did not change. The wormhole check does not require a large overhead. Thus in a system with fast mobility, DeWorm can be used every time a source wants to communicate with a destination and suspects that it may be communicating through a wormhole.

3.6.2 DeWorm and Routing Protocols

So far we have not discussed using DeWorm with any specific routing protocol. A comprehensive survey of routing protocols for ad hoc networks is presented in [14]. DeWorm is designed to be universal and work with any routing protocol. However, some proactive routing protocols may provide useful and ready information that can be used by DeWorm. For example, the information needed in steps 1 to 4 of DeWorm, described in Section 3.3, may already be available in the routing table. If this information is fresh, then the nodes can use it to find the shortest route to the target node that avoids the blacklisted neighborhood nodes. This will reduce the overhead and delay with DeWorm. Other routing protocols may need some modifications to work with DeWorm (e.g., those designed to provide nodes only with the next hop that will deliver their packets to the destination and not the entire route).

The idea of finding routes that avoid blacklisted nodes have been used by other protocols proposed previously in the research literature [57]. Routing protocols are run on top of such protocols to update and distribute the blacklist. Since we need the nodes to find routes to the target node which is normally 2 or 3 hops away, a node can also simply broadcast its request to its neighbors to find the shortest route to the target node that avoids certain nodes. We used this simple broadcast approach in our performance evaluation in Section 3.9.1. Analysis in Section 3.9.1 and simulations in Section 3.9.3.7 show that the average number of route acquisitions is small, which means broadcasting 2 or 3 messages to direct neighbors to find the routes is not an expensive approach.

In DeWorm, the nodes on the route from the source to destination check for the existence of a wormhole. This assumes that the source already knows all the nodes on the route to the destination or at least the next two hops to the destination. However, some proactive mobile ad hoc network routing protocols are designed to provide nodes only with the next hop that will deliver their packets to the destination. The Destination Sequence Distance Vector (DSDV) [58] which is based on the traditional Bellman-Ford algorithm is a common proactive mobile ad hoc routing protocol. In DSDV the routing tables contain: an entry that stores the next hop towards a destination, the cost metric for the routing path to the destination and a destination sequence number that is created by the destination. This means that the sender knows the next hop that will lead him to the destination. The sender will have to ask the immediate next hop to provide him with his next hop node on the route to the destination. Then the Sender can ask his one-hop neighbors to target that node. Another solution is to ask all the one-hop neighbors to target the destination node and provide the sender with the length of their routes. As mentioned previously, we have tested, using simulations, a fixed target node for all nodes (we fixed the destination node as the target) and the results showed that DeWorm can still work efficiently.

On the other hand, with Dynamic Source Routing (DSR) [59] each node learns the complete route information and uses caching technology to maintain them. With DeWorm, if the sender node has the routing information to the destination available in its cache then it will immediately inform its neighbors about the target node. Otherwise, if the routing information is not available then the sender will send a route request and wait for a route reply.

3.7 WORMHOLE REMOVAL

In Chapter 4 we will propose a wormhole removal process. DeWorm can also adapt the removal process so that it will not only detect the existence of wormholes in the route from a source to a destination, it can also help prevent nodes from using the wormhole link. If a node in DeWorm detects a wormhole then it will ask all its neighbors to use the removal process with all their neighbors. In [4], a removal scheme was presented that could remove legal links while removing all illegal links containing the wormhole.

3.8 SIMPLIFIED DEWORM

In this section we propose another version of DeWorm called Simplified DeWorm. It is more suitable for applications (e.g., sensor networks) where we need to decrease the number of nodes that are involved in wormhole detection. Such networks could be static and/or involve nodes that employ management protocols (e.g., sleep) to control the energy consumption and increase the lifetime of the network. In Simplified DeWorm we only change step number 4 of the DeWorm protocol. Here, the sender will randomly pick a node from the list of one-hop neighbors excluding the marked neighbors and the previous and next hops along the route. The randomly picked node has to find a route that does not pass through any of the one-hop neighbors of the sender. In order for the detection algorithm to work we need to ensure that the randomly picked neighbor is not located on the other side of the wormhole (e.g., node B should not randomly pick node 8 to compute a route to E). If so, the computed test route to the destination will be short like the current route through the wormhole, and the wormhole attack will not be detected.

When a node that is within the transmission range of M_1 sends a “Hello” message, M_1 will transmit it to M_2, which will replay this message on the other side of the network. Therefore the node will receive replies from nodes located on the other side of the network. All these replies will also be replayed by the same transmitter M_1. One can expect that the signals of these replies will likely have very similar RSS values when received by the sender of the “Hello” message. This feature can be used to reduce the chance that the sender will pick a node located near M_2 to compute a route to the target node. The sender now excludes nodes that replied with RSS values that are similar or those that are very close to the RSS value of the node along the route. We have not evaluated what “similar” RSS means here and this is beyond the scope of this work. Measurements of RSS values in 802.11 networks show that they are approximately normally distributed [60] and although the RSS varies in time, in a time period as large as 15 minutes, the average value of the RSS from a sender is relatively stable. One possibility is to monitor the average RSS value over packets and decide if two RSS values from two different nodes are similar. Of course there is a finite probability that they are different.

The wormhole will have transceivers that can detect the signals at one side of the network and transmit them at the other side of the network. In simplified DeWorm, we have assumed that all the signals that are transmitted consecutively will have similar signal properties and thus produce similar or very close average RSS values. Signals that have different RSS values are thus assumed to be from different transmitters. If the transceiver of the wormhole can send signals with different transmit powers, the protocol will not guarantee the detection of a wormhole.

In any case, since nodes that determine the alternative routes are picked randomly, there is a still a finite chance that the node is picked from the set of legitimate one-hop neighbors that are located within the transmission range of the node, and not on the other side of the wormhole. This probability can be increased if more than one neighbor is picked.

3.9 PERFORMANCE EVALUATION

3.9.1 Overhead Analysis

Here we discuss the overhead resulting from the use of the DeWorm protocol. We employ a model with given network specifications (such as: network size, number of nodes, and nodes’ transmission range) to determine the following: (i) The average number of packets that need to be broadcast by the sender nodes (these contain information about the target node and the forbidden list of nodes). (ii) The number of route acquisitions performed by neighbor nodes to the target node, which also equals the total number of replies (contains information on the length of the route to the target node).

We start with the number of nodes that need to become "senders" to check for the wormhole until it is detected. This equals the number of broadcast messages and this number depends on the number of hops between the sender and the destination and the position of the wormhole. Let us suppose that nodes are uniformly and randomly distributed in a square area of size A^2. Nodes can communicate directly if the distance between them is less than the transmission range R. Let $d_{i,j}$ be the distance between two nodes i and j. Let $N_{i,j}$ be the number of hops of the shortest path between nodes i and j. Then we have, as shown in [21], the minimum number of nodes between the sender S and the destination D as: $N_{S,D} \geq d_{S,D}/R$ and, $N_{S,D} = \beta d_{S,D}/R$, where $1 \leq \beta \leq 2$. A proof is shown in [21].

Let M_1 and M_2 be the transceivers of the wormhole located somewhere between S and D. The wormhole will be detected when the latest "sender" along the route is located within M_1's range. In the best case M_1 could be a neighbor of S and thus detected immediately. In the worst case M_1 could be two hops away from D (M_2 is D's neighbor – we assume that the special case in Section 3.3 is identical to other cases but it requires a few extra messages). Thus, on average, the number of "senders" that need to check for the wormhole will be:

$$N_{Check} = (\beta d_{S,D}/R - 2)/2 = (\beta d_{S,D}/2R) - 1$$

With a square area of size A^2, the longest distance between S and D can be $\sqrt{2}A$. This happens when the sender and destination are located at two opposite corners diagonally. The maximum number of sender nodes that need to check for the wormhole before the wormhole is detected N_{Check} is:

$$N_{Check} = (\sqrt{2}\beta A/2R) - 1 \tag{3.1}$$

The probability of having k number of neighbors within the transmission range R of a node can be derived as in [61]:

$$P(k) = \begin{pmatrix} N \\ k \end{pmatrix} (\pi R^2/A^2)^k (1 - \pi R^2/A^2)^{N-k}$$

where N is the total number of nodes in the network. Thus the average number of neighbors will be:

$$AV_k = \sum k \cdot P(k) = N \cdot \pi \cdot R^2/A^2$$

The expected number of replies AV_{Rep} that each sender will receive will be equal to average number of neighbors excluding the previous and the next hop along the route to the destination node. The total number of replies and also the total number of route acquisitions to the target node will be:

$$N_{Check} \cdot AV_{Rep} = ((\sqrt{2}\beta A/2R) - 1) \cdot (\pi \cdot N \cdot (R/A)^2 - 2) \tag{3.2}$$

For a given A, R, and N, the number of packets that need to be broadcast by sender nodes is given by (3.1). We show that this is not a significant overhead in section 3.9.3.7. Later in section 3.9.3.7 we will use the number of route acquisitions given by (3.2) with the parameters of the simulated network and $\beta = 1$ and $\beta = 1.5$, to define a lower and upper bound for the number of route acquisitions. This will be compared with the actual number from simulations and shown to be a reasonably accurate.

3.9.2 Simulations

The effectiveness of our protocol to detect wormhole attacks is evaluated in this section using extensive simulations. The important metrics for wormhole detection are: the percentage of correct detection of the wormhole and the percentage of false positives. We evaluated these metrics for various node distribution and connectivity models. We used distributions and connectivity models similar to those used in [4]. In our simulations we considered two connectivity models: the unit disk graph and the quasi-unit disk graph. Two different node distribution models were used in the simulations: grid distribution with some perturbations and random distribution. Finally, we studied the performance of our protocol in the case of asymmetric links, again with different connectivity and node distribution models.

Connectivity and distribution models: The unit disk graph model (UDG) [52] is widely used for studying ad-hoc networks. In this model the transmission range of nodes with omni-directional antennas is modeled as a disk of unit radius. Links between a node and its neighbors will exist only if they fall within the disk. The UDG model does not really represent reality because it does not consider the vagaries of radio propagation [52]. In the Quasi-UDG model [52], a link between two nodes will exist if the distance between them is less than αR, where R is the transmission range of the node and α is the quasi-UDG factor (where, $0 \leq \alpha \leq 1$). We used $\alpha = 0.75$ in our simulations. A link will not exist if the distance (d) is greater than R. However, the existence of a link is not specified if $\alpha R \leq d \leq R$, thus for this case we assumed that the link will exist with probability $(R - d)/(R - \alpha R)$.

In our simulations we considered 144 nodes, distributed in a 1200m $\times$ 1200m square area. To change the average node degree, the transmission range of the nodes was varied from 120m to 160m. As we have discussed earlier and as shown later by the simulation results, our protocol does not require high connectivity for good performance.

For the random node case, the coordinates of the nodes (x_i, y_i) for $i = 1, 2, ...144$ were independently and randomly chosen in the range from 100 to 1200m using a uniform [100-1200] random number generator. In the grid case, nodes are located in a perturbed 12×12 grid. The coordinates of each node x_i and y_j were randomly chosen using uniform random variables in the ranges $(100i - p100, 100i + p100)$ and $(100j - p100, 100j + p100)$, respectively, where p is the perturbation parameter and $i = 1, ...12$ and $j = 1, ...12$ (in our simulation we choose $p = 0.2$). After the nodes are distributed, the connectivity model (UDG, quasi-UDG) and the transmission range determine the network topology.

To change the average node degree, the transmission range of the nodes was varied from 120m to 160m. After the nodes are arranged in the network according to the distributions and connectivity models, the sender node is randomly chosen from the left-most nodes in the network (nodes with $x < 200$) and the destination node is randomly chosen from the right-most nodes (nodes with $x > 1000$). The wormhole is randomly created somewhere between the sender and the receiver with a random length that is uniformly distributed between the nodes' maximum transmission range to little less than the distance between the source and the destination.

For each combination of connectivity and distribution models and sensitivity parameter values the simulation is repeated with different node distributions, sender-destination pairs, and wormhole lengths and locations. The percentage of detection is the number of times

DeWorm successfully detected the the wormhole out of the total number of times the simulation was run (1000 times). We also measured the percentage of false positives, which occurs when a node in the route to the destination that is not within transmission range of the wormhole transceiver mistakenly detects a wormhole. Running and testing the effectiveness of DeWorm only requires routing information. Since no other traffic is simulated, simulating DeWorm in C is much faster compared to ns-2 or OPNET. Simulations were programmed in C using routing and node distribution models from ns-2 and were run for more than 1000 times for each case with different node distribution and wormhole positions for each run.

3.9.3 Results

3.9.3.1 Method for Selecting the Route for Comparison In Section 4.3.1 we discussed various possible methods for selecting the alternate route whose length L is used by DeWorm to detect the wormhole. Table I shows the percentage of wormhole detection and false positives for various selection methods: comparing the current route with the average length of all routes, the longest route, and using the sensitivity parameter to eliminate the longest routes, but compare with the next longest as described in Section 4.3.1. We show the results for both grid like and randomly distributed networks and with UDG connectivity. We only show the results for a sensitivity parameter of 3. The results show that using the sensitivity parameter as described in Section 4.3.1 provides the best detection percentage and a reasonable number of false positives. Thus in all our simulations we used this method for selecting the route for comparison. The results show that using the longest route is a very sensitive method that can efficiently detect the short wormholes. However, it causes a high number of false positives, especially in random networks. This is because it will pick the longest route from a neighbor to the target node, which could be 2 or more hops longer than other routes that avoid the wormhole.

3.9.3.2 Detection and False Positives In this section we will show the percentage of detection and false positive for the DeWrom protocol, which are the most important metrics for any wormhole detection protocol. The network is simulated in this section with

	Grid		Random	
	Detection	False positives	Detection	False positives
Average	98.5	0.1	98.2	0.31
Sensitivity	99.63	0.4	99.65	1.42
Longest	100	1.65	99.7655	8.9

Table 3.1: Comparison of route selection method

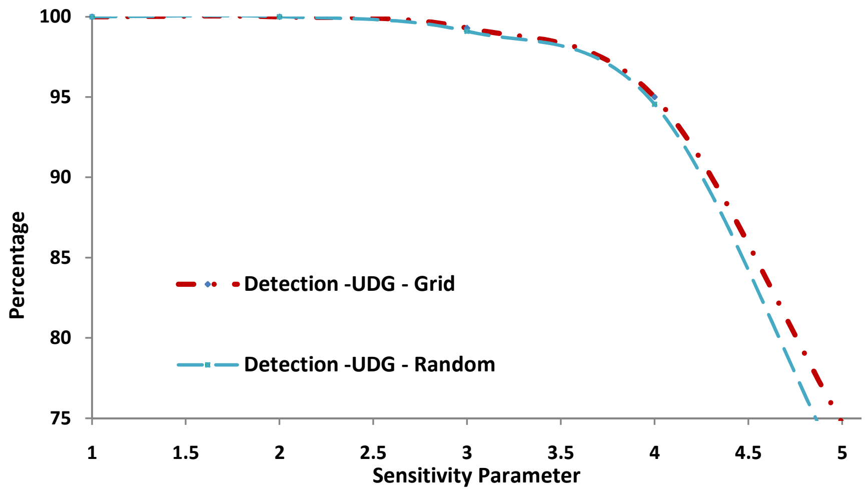

Figure 3.4: DeWorm Detection with UDG

symmetric links and with two connectivity models: UDG and Quasi-UDG.

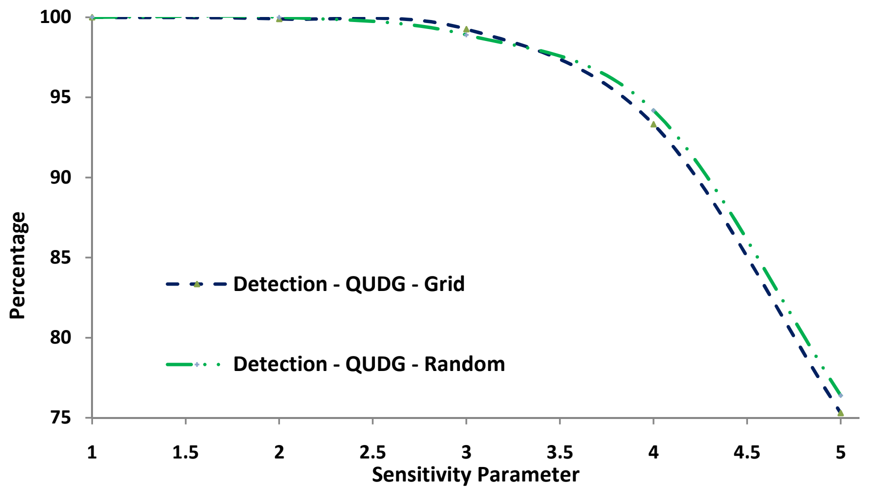

Figure 3.5: DeWorm Detection with Quasi-UDG

Figure 3.4 shows the percentage of wormhole detection with DeWorm for a grid distributed network with $p = 0.2$ and randomly distributed networks, with various values of sensitivity parameter. We see excellent results for both distribution models. In both cases, the detection is almost 100% for sensitivity parameter values of 2 and 3. Almost no false positives occurred. Figure 3.5 is similar to Figure 3.4 but with Quasi-UDG connectivity model. The results show that the percentage of detection is effeceted and DeWorm can still detect wormholes effectively with network with Quasi-UDG connectivey models.

Figure 3.6 shows the percentage of false positives for grid and randomly distributed networks with UDG connectivity. With a sensitivity parameter of 3, there are very low number of false positives. However, in the random distribution case, the effect of the sensitivity parameter is more significant (higher false positives for a sensitivity of 2). This is due to the randomization in the node distribution. Thus the routes from the neighboring nodes may be longer than the direct route, with the difference being greater than a small sensitivity parameter. This will result in more false positives. Figure 3.7 is similar to Figure 3.6 but with Quasi-UDG connectivity model.

From the results in this section it can be observed that as we increase the sensitivity parameter, the percentage of false positives decreases. The increase in the sensitivity param-

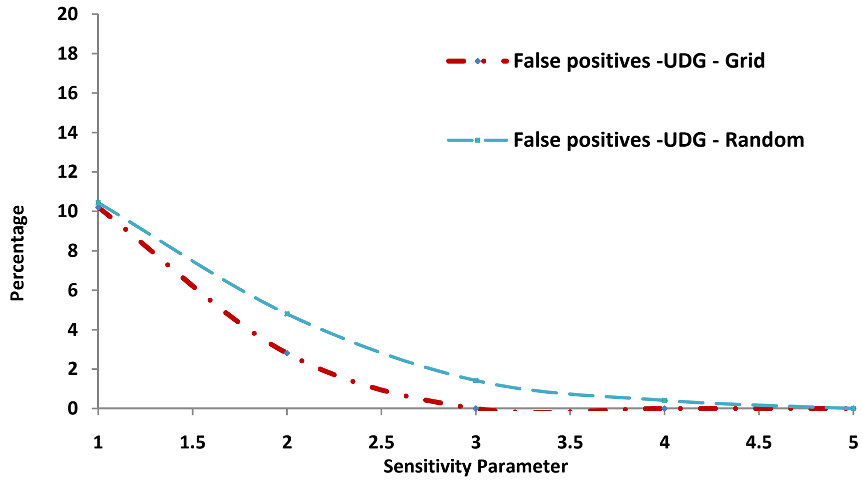

Figure 3.6: DeWorm False Positives with UDG

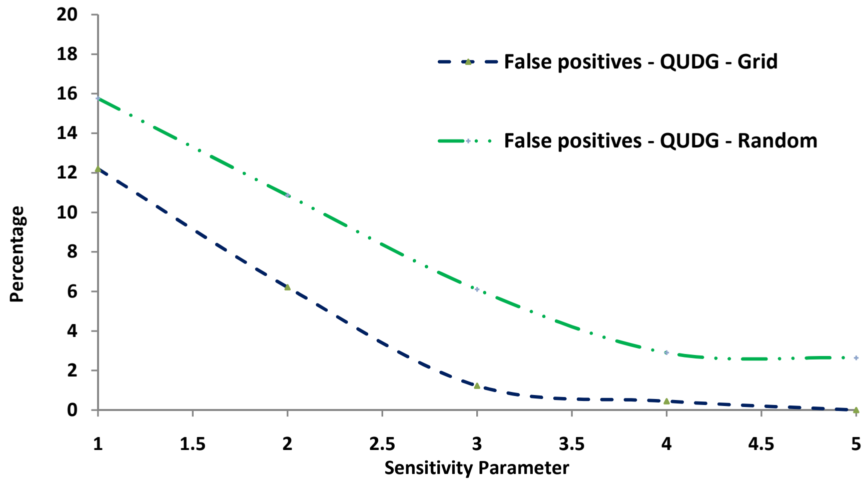

Figure 3.7: DeWorm False Positives with Quasi-UDG

eter will however reduce the detection rate of short wormholes. Nevertheless, the results are very positive. Using Quasi-UDG will have a small impact on the results. This is because the nodes will be reaching less neighbors with the Quasi-UDG and thus may find longer routes to the target nodes causing false positives.

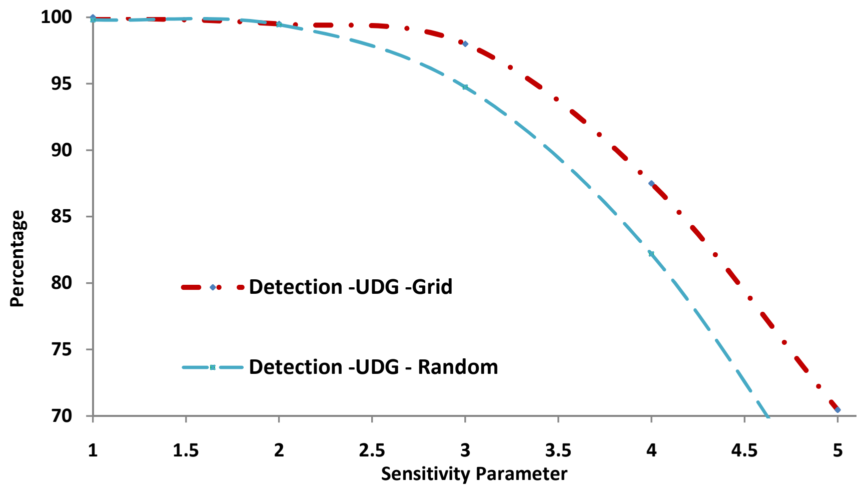

Figure 3.8: DeWorm Detection with Asymmetric UDG

3.9.3.3 The effect of Asymmetric links We use an asymmetric link model similar to the one in [62]. The transmission range of a node is determined by the power level of the node. Nodes are classified into high power nodes with maximum transmission range, and low power nodes with minimum transmission range (equal to half the maximum range). Figures 3.8 and 3.9 show the percentage of wormhole detection with UDG and Quasi-UDG and asymmetric links for grid-like and randomly distributed nodes, respectively.

Like the symmetric case, with sensitivity parameters of 2 and 3, the wormhole detection is almost 100%. DeWorm is thus capable to detect womrmholes even in the presence of asymmetric links.

Figures 3.10 and 3.11 show the percentage of false positives of DeWorm with UDG and Quasi-UDG and asymmetric links for grid-like and randomly distributed nodes, respectively. Compared to the symmetric links, results with asymmetric links the percentage of false positives is slightly increased. This is because with asymmetric links some nodes will have

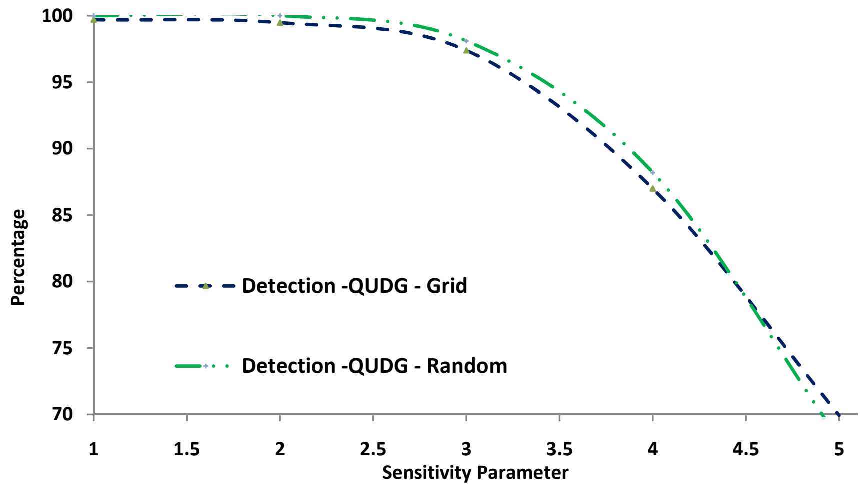

Figure 3.9: DeWorm Detection with Asymmetric Quasi-UDG

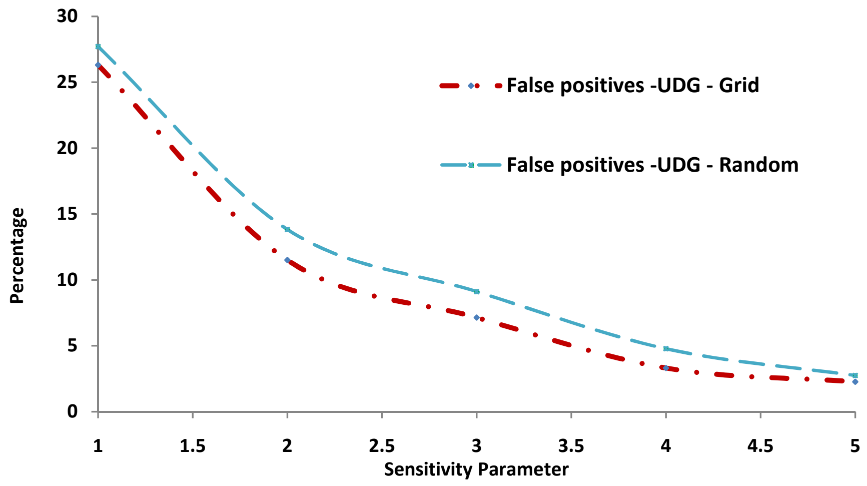

Figure 3.10: DeWorm False Positives with Asymmetric UDG

a short transmission range and could not reach many neighbors. Thus they may have long routes to the target node causing false positives.

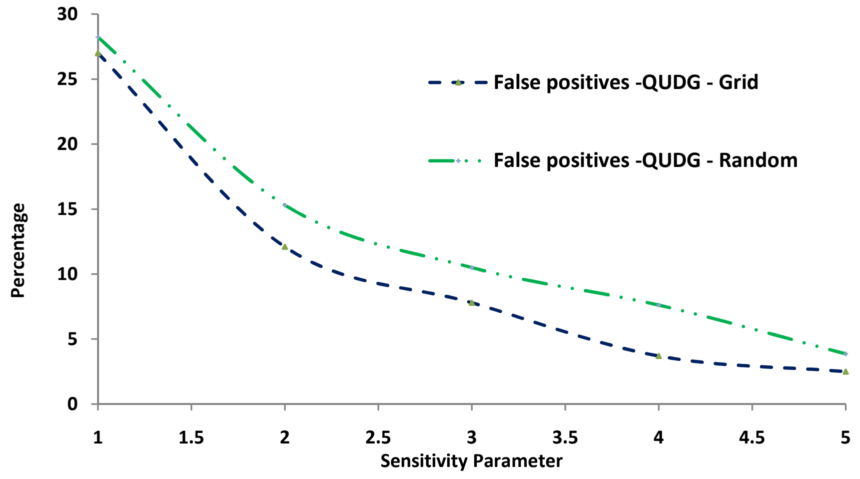

Figure 3.11: DeWorm False Positives with AsymmetricQuasi- UDG

In our simulations, 50% of the nodes randomly have half the transmission range. However, we had to increase the transmission range by a small margin to maintain connectivity in this case. In the literature there is no clear definition of the node degree for networks with asymmetric links.

3.9.3.4 The effect of average node degree The protocol in [4] requires a node degree greater than 7 to achieve a 100% detection rate with random node distribution. Our protocol does not require such a large node degree for similar performance. Figures 3.12 and 3.12 show the detection percentage and percentage of false positives for various node degrees with grid and random node distributions, respectively. In the simulations, the average node degree was changed by changing the transmission range of the nodes and by changing the network size. The larger the transmission range and the smaller the network size the higher is the node degree. Increasing the node degree by more than 5 makes no difference in the detection/false positive percentages (for this reason, in the other results, we use a fixed node degree of 5 and 6).

The results show that DeWorm will still work for a low average node degree and achieves

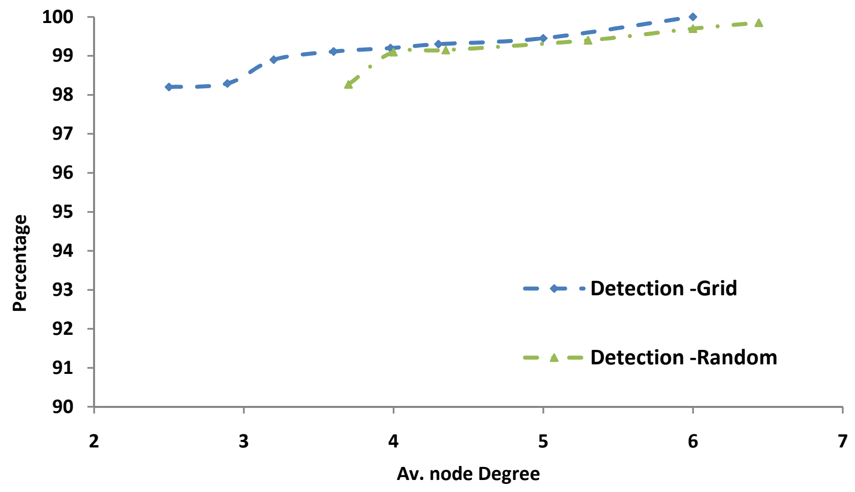

Figure 3.12: Impact of node degree on the Detection of DeWorm

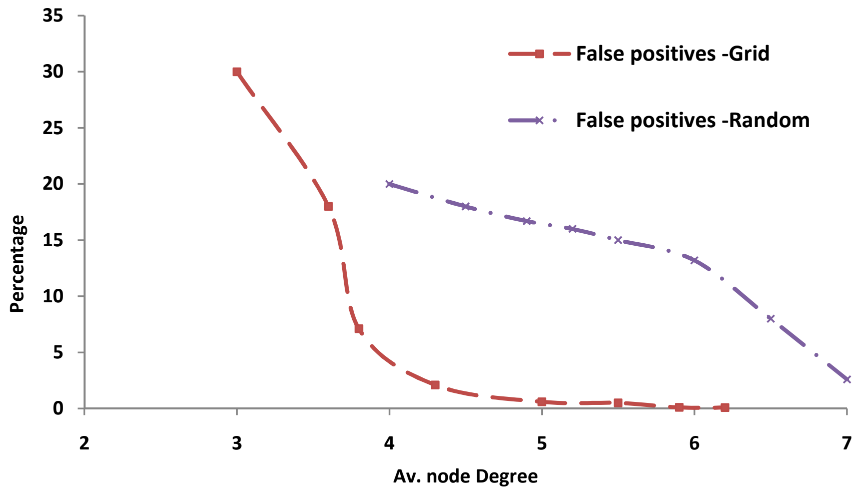

Figure 3.13: Impact of node degree on the Detection of DeWorm

excellent detection rates. However, the percentage of false positives is high for node degrees less than 4. The underlying reason for this is that fewer links are available between the nodes, and thus, the routes from the neighboring nodes may take physically circuitous paths and result in much longer routes (in terms of hops) to the target node. In Figure 3.12, we only show the detection probabilities for average node degrees higher than 3.7 for random node distribution. The reason is that for node degrees less than 3.7, the network will not have enough connectivity and sometimes the neighboring nodes will not have routes to the target node.

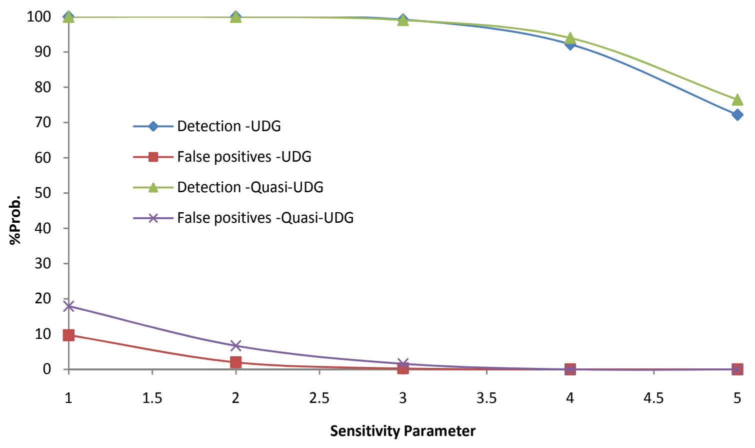

Figure 3.14: simplified DeWorm

3.9.3.5 Simplified DeWorm Simplified DeWorm was simulated to evaluate its performance. The route to target node was compared with the length of the route delivered by a randomly picked neighbor after excluding marked nodes. We do not simulate the RSS values, but automatically include neighbors on M_2's side of the wormhole in the marked list. Figure 3.14 shows the percentage of detection and false positives with simplified DeWorm. For brevity, we only show one case for a grid distribution and with UDG and Quasi-UDG connectivity models with symmetric links. Simplified DeWorm performs as efficiently as DeWorm. We expect it to perform well under other scenarios as well.

Wormhole length	1.5R	2.0R	2.5R	3.0R	3.5R	4.0R
detection-Grid	94.0	99.48	100	100	100	100
detection-Random	80.3	89.22	93.1	98.55	99.56	100

Table 3.2: Impact of wormhole length

3.9.3.6 The Impact of Wormhole length We tested DeWorm with different wormhole lengths. We ran 1000 simulations with Q-UDG, symmetric links, $R = 150m$, $\eta = 2$, and for both grid and random topologies. The results in Table 3.2 show that DeWorm can detect short wormholes (1.5R) nearly 95% of the time for grid and 80% of the time for random topologies.

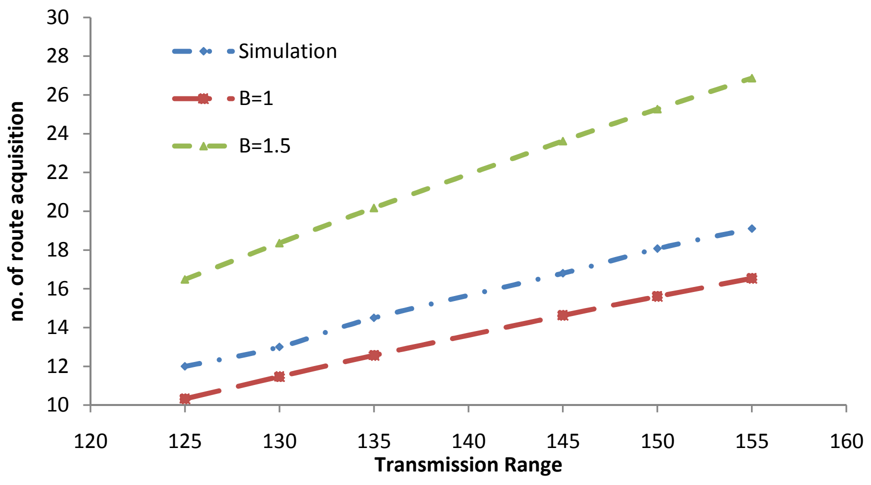

Figure 3.15: Number of route acquisitions

3.9.3.7 Overhead Figure 3.15 shows the total number of route acquisitions with different R. We used (3.2) with $A = 1200$m, $N = 144$, and R was varied from 125m to 155m. We chose $\beta = 1$ and $\beta = 1.5$ to obtain the lower and upper bounds for the total number of route acquisitions, respectively. In the simulations, we considered the grid-like distributed nodes case with the UDG connectivity model. We averaged the total number of route acquisitions for 1000 runs. The results falls between the lower and upper bounds from (3.2) verifying its accuracy. The number of route acquisitions increase as the transmission range is increased because the number of neighbors of the node increases. Thus, more nodes will send route acquisitions to the target node. Our simulations also showed that for $R = 135$ the average number of messages that need to be broadcast is 3.5. Using the same parameters in equation (3.1) will result in 3 and 5 as the lower and upper bounds for the number of messages that need to be broadcast. All of these numbers are small and thus DeWorm needs very minimal overhead especially as it is used only on demand.

3.9.4 Comparisons to other detection protocols

Here, we will mainly focus on protocols that do not use location information, time measurements, or special hardware. The detection method in [4] only relies on connectivity information. As we discussed earlier, we used the exact same setup and models that were used in [4]. However, their approach does not rapidly adapt to topology changes and it is more suitable for a static network. Their approach requires a high degree of connectivity to achieve good performance, especially in the case of random distributions of nodes. For example, for random node distribution and quasi-UDG connectivity model, results in [4] for an average node degree around 4 showed that the detection percentage was between 50-65%. This can be compared to 99.1% detection using DeWorm for a similar case. Also we have high detection rates for lower node degrees (99.15% for average node degree of 2.5 with grid, and 98.27% for an average node degree of 3.6 with random node distributions).

3.10 OTHER WORMHOLE ATTACKS

In this section we will propose two wormhole attacks: the smart wormhole attack and the multiple wormholes attack. We will show that the DeWorm protocol in its current state is not able to detect these attacks. Thus, we will propose a minor modification on the DeWorm protocol to detect these advanced wormhole attacks.

3.10.1 The Smart Wormhole Attack

It is possible that transceivers M_1 and M_2 are so advanced that they can advertise different neighboring list for nodes in their vicinity. This attack is beyond the known wormhole attack addressed in the literature [29, 27]. We call this new attack the smart wormhole attack.

Let us suppose that transceivers M_1 and M_2 can now filter nodes they allow to connect through the wormhole link and select nodes either randomly or in an intelligent manner with time. They can accomplish this by changing their transmission/reception range (this will limit or increase the number of nodes covered by the wormhole transceivers) or by selectively picking "hello" messages and replies to relay. Thus when M_1 receives "hello" messages from nodes within its transmission range, M_2 will selectively decide which ones to replay to its neighbors. Also M_1 will selectively decide which replies to broadcast for each "hello" message. The idea is not to advertise all the nodes that are within M_2's transmission range and hide some of them. Thus, these nodes will be excluded in the list of nodes that a "sender" will send to his neighbors to exclude in their routes to the target node. Later, when the neighbors try to find a route to the target, M_1 can advertise one of the previously hidden nodes as a neighbor with a short legal route to the destination. This may deceive the DeWorm protocol.

The smart wormhole attack is also capable of thwarting other wormhole detection protocols. For example, if the detection protocol in [4] is used, then the wormhole can adjust its transmission range to cover only one node. In this case there will not be sufficient numbers of independent neighbors to detect the wormhole attack. This limitation of [4] was also discussed in [37].

We suggest a minor modification so that this cannot thwart DeWorm. The "hello" messages are encrypted so that M_1 and M_2 will not know which node sent the "hello" message. This will prevent the situation where M_1 and M_2 can collaborate to selectively decide to hide and advertise messages from a specific neighbor. The sender node can try to find his one-hop neighbors again immediately after he receives the route information from his one-hop neighbors. If the list is changed, then the sender will broadcast the new list also to incorporate this information during execution of DeWorm. With this action the wormhole transceivers will be prevented from advertising different neighbors to the neighbors of a node. Thus, cannot deceive the DeWorm protocol. It is also possible to make the neighbors not only reply to the sender node with the length of their routes to the target node, but also with the nodes they used to reach the target node. The sender will recheck its neighbors to see if new nodes have been added to its neighbors list then it has to ensure that none of the new nodes is used by its neighbors to reach the target node. Otherwise, the sender will ask the neighbor that that used a new node to find a new route avoiding the new neighbors in the list.

3.10.2 The Multiple Wormholes Attack

We also discuss the multiple wormhole attack here. In this case the attacker will deploy multiple pairs of transceivers to form multiple wormhole links. Implementing this attack is easier than the smart wormhole as it does not require advanced transceivers. The goal of the attacker here is have the neighbors of the node go through the second wormhole even as they avoid the first wormhole.

An example of a multiple wormhole attack is shown in Figure 3.16. Note that the transceivers of the wormholes cannot overlap, otherwise nodes in their range will be included in the neighboring list and will be avoided. That is, the two wormholes will present a single wormhole with an extended transceiver's range. In this example, node A will ask its neighbors nodes K, N, D, X, and F to find routes to the target node T. At least nodes K, N and D are supposed to find long routes as they have to void nodes X, F and B. Note that nodes P and Z are within the range of the blue wormhole and not the red wormhole. Thus, they will not be avoided by the neighbors of node A. For example, node K will only avoid the red wormhole but not the blue wormhole. The question now is whether DeWorm can still detect the existence of a wormhole? A route from a neighbor of node A, node K, will at least require the following nodes to work:

1. A node (O in this example) that will reach a node that can connect a node from one wormhole transceiver range (M_1) to another node from the other wormhole transceiver range (M_3).

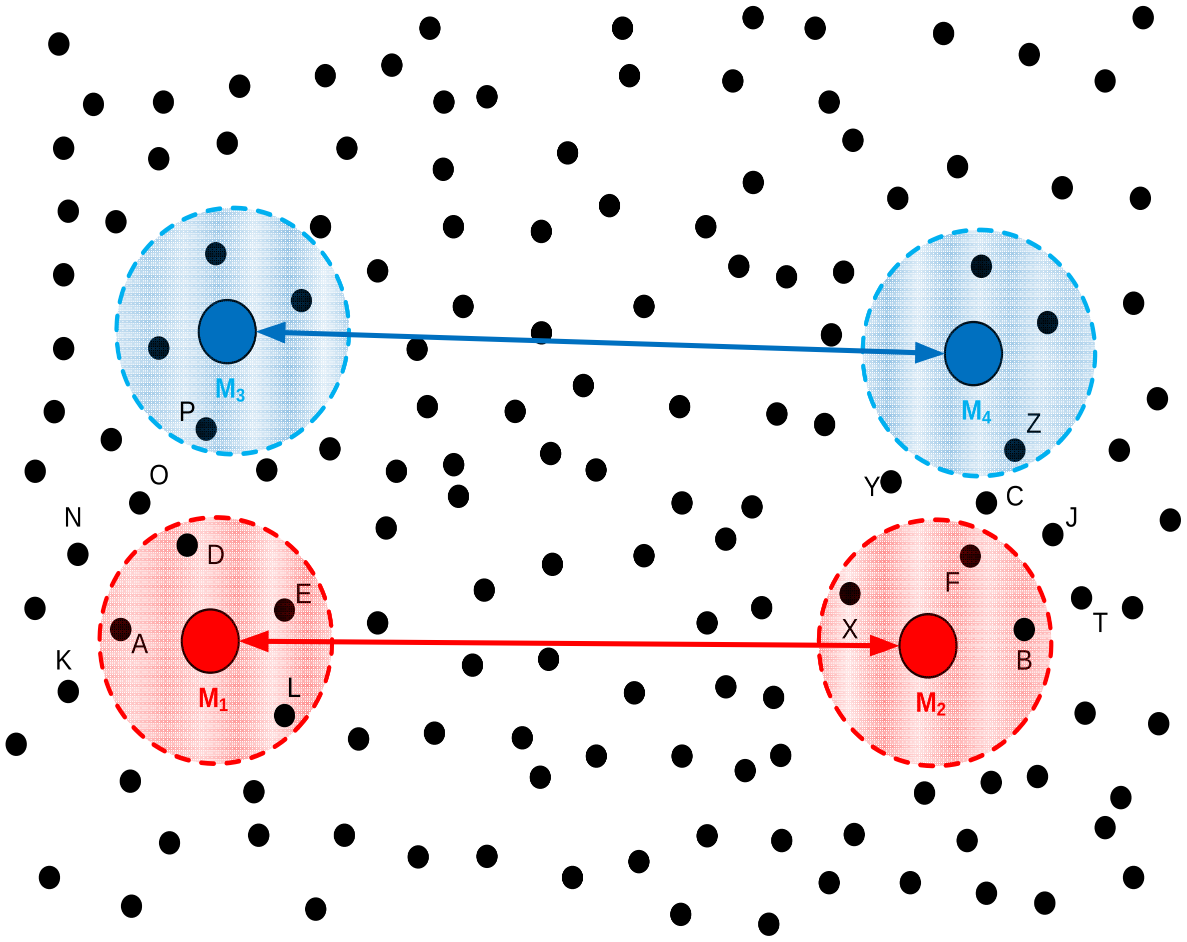

Figure 3.16: Parallel Wormhole Example

2. A node that is within the range of the transceiver M_3, node P in this example.
3. A node that is within the range of the other side of the wormhole M_4, node Z in this example.
4. A node that is outside the other side of M_4 range, node C in this example.

Thus, a route that will avoid the red wormhole and go through the blue wormhole will be at least 4 hops longer than than the direct route. Consequently, DeWrom with a sensitivity parameter ≤ 4 will be able to detect any multiple wormhole attack. In order to avoid the situation where the intermediate node is not needed and a node that is within M_1's range directly communicates with a node within M_2's range the neighbors of the "sender" must also use the DeWrom protocol to reach the target node. In Chapter 4 a network with two wormholes and with multi-end wormhole will be simulated and tested.

4.0 SECURE NEIGHBORHOOD CREATION

4.1 INTRODUCTION

Nodes in ad hoc networks try to discover their neighbors simply by broadcasting a neighbor discovery request. Each node that hears the request responds with a neighbor discovery reply. An adversary may try to thwart neighborhood discovery to disrupt the network operation by (a) preventing two neighbors from discovering each other by jamming or (b) creating a "neighbor relationship" between nodes that are not really in range of each other. The latter can be accomplished by spoofing neighbor discovery messages or by installing wormholes in the network. Cryptographic techniques (authentication and encryption) can often prevent the adversary from messing with the discovery messages. Wormhole attacks, considered here, cannot be addressed using cryptography. Jamming attacks are not considered in this work.

As discussed in the previous chapters an adversary can easily construct a wormhole by simply copying all packets (signals) from one location (near M_1) in the network and replaying them at another location (near M_2) that is located several hops away. Then all the replies and packets (signals) from the location near M_2 will also be captured and replayed at the location near M_1. Since the adversary can capture the signals or bits, cryptographic techniques will not help to prevent or detect this malicious behavior. As a result of this attack, nodes (from 1 to m_1) that are located in M_1's area will believe that they are neighbors to nodes (from 1 to m_2) that are located in M_2's area. Obviously, this will impact the operation of a neighborhood creation protocol. This can logically be seen as a single two-end wormhole, but physically it has created $2 \times m_1 \times m_2$ bogus links between nodes in the network. Note that a node A near M_1 will believe it is a neighbor of node B near M_2 and vice versa. It is not sufficient to flag the link from node A to node B as bogus. It is also necessary to flag the link from node B to node A as bogus. Otherwise, node B may continue to assume that node A is its real neighbor.

A short survey of neighborhood discovery in ad hoc networks in provided in [37]. This paper also provides definitions of neighborhood types and neighborhood discovery protocols. In this paper, the authors conclude that securing neighborhood discovery is a difficult and open problem. The proposed protocols described in Chapter 2 require that the detection protocol be applied between all nodes and all the their neighbors to to detect the existence of a wormhole. In large networks with high node degrees, this will cause a significant overhead and delay.

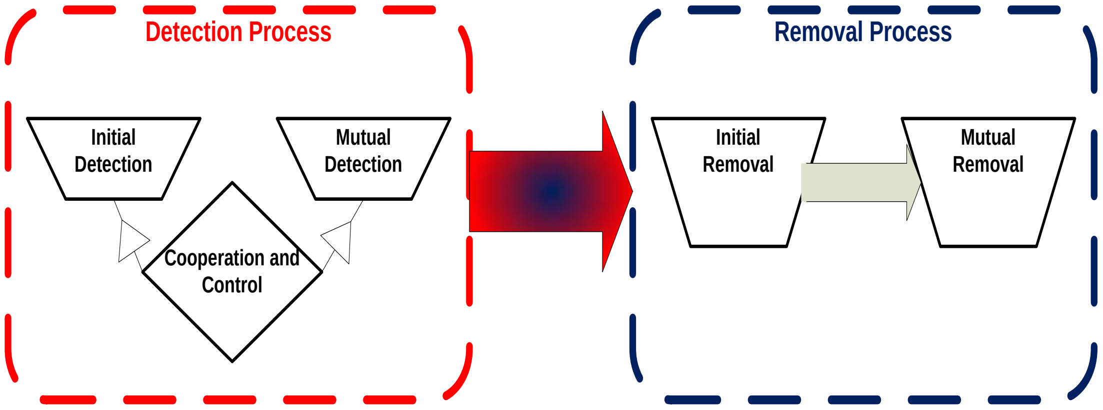

Figure 4.1: Secure Neighbor Creation Protocol

In this chapter we propose a secure neighborhood creation protocol that can securely discover the neighbors of a node in mobile ad hoc network by first detecting wormholes, if they exist, and then removing bogus direct links between nodes. Compared to other secure neighbor verification or discovery protocols, our protocol is simple, localized, and needs no special hardware, localization, or synchronization. The protocol can also detect and remove multiple two-ended and multi-ended wormholes.

We describe the secure neighborhood creation protocol at a high level. As shown in Figure 4.1, the proposed secure neighborhood creation protocol consists of two main processes: *the detection process* and *the removal process*. The detection process itself incudes three operations: initial detection, mutual detection, and co-operation & control rules. The initial detection operation, will provide near 100% detection for different types of wormholes, but unfortunately also results in a high number of false positives. The mutual detection operation, will enhance the performance of the initial detection by still maintaining a high detection rate of wormholes but significantly reduces the number of false positives. The co-operation and control rules will control the use of the previous two stages (instead of having all nodes applying initial detection to all their neighbors, this stage will define the rules that each node will follow) and will reduce the overall overhead significantly. However, the process still maintains the high detection rate and results in a very few number of false positives.

The second process in the secure neighbor creation protocol is the *removal* process. This process is the most complicated stage and depends on the results produced by the previous stage. It consists of two operations: initial removal and mutual removal. If there are no wormholes existing in the network, the removal process is supposed to not remove any link by mistake. However, if there is a wormhole then it must successfully remove all links between nodes connected using the wormhole. The mutual removal operation will enhance the performance of the initial removal operation by reducing the number of links removed by mistake.

To our knowledge this is the first protocol that provides co-operation rules between neighboring nodes to reduce the overhead that may be caused if all nodes in the network need to verify all their neighbors. Also this is the first protocol that can provide 100% removal of all the wormhole links and only remove very few, if any, legal links.

Our simulations and analysis show that the proposed protocol can successfully detect and remove all wormhole links. Very few false positives will occur and very few legal links will be removed by mistake. The cost, as will be evaluated, is very low overhead measured by the number of route acquisitions.

The rest of this chapter is organized as follows. At first we will start with a detailed description of all the processes and operations of the secure neighbor creation protocol. This will be followed by a discussion of some related issues for the proposed protocol. Finally, a comprehensive evaluation using simulations will be presented.

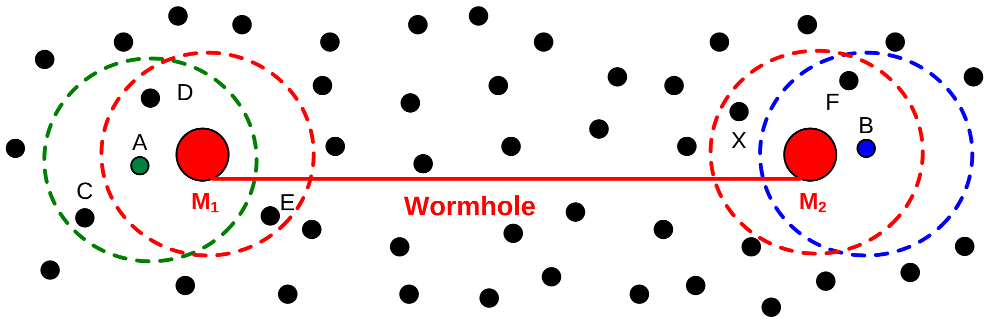

Figure 4.2: Neighborhood Creation Problem

Figure 4.2 shows a network with a wormhole connecting two sides of the network. All the nodes that are inside the two red circles (with dashed perimeters) are connected and they are not aware of this fact. In the example shown in Figure 4.2, node A will try to discover its one hop neighbors. Node A will not only receive replies from its actual one hop neighbors, nodes C and D, located inside the green circle, but also will receive replies from nodes B, F, and X connected by the wormhole. In this case the operation of the neighborhood creation protocol iss disrupted and nodes A, E, and D are fooled into believing that they are neighbors of nodes B, F, and X, and vice versa. The main goal of any secure neighbor discovery protocol is to detect the existence of the wormhole and if possible remove *all* of the fake links created by the wormhole. In fact the removal process is complicated as it must not remove legal links – for example links $A - C$ and $A - D$ must not be removed.

As shown in Figure 4.1, the proposed secure neighborhood creation protocol consists of two main processes: *the detection process* and *the removal process*. The goal of the detection process is only to detect the existence of a wormhole in the vicinity of a node checking for it. For example, node A may detect that there is a wormhole connecting it with one or more neighbors. This process is not responsible of pinpointing the bogus links created by the wormhole. The result of the detection process will feed into the removal process. Only if something fishy is detected by the detection process, then the removal process will be employed. This implies that the detection process must almost always detect wormholes, if they exist. Otherwise wormhole links may not be removed. Also, the detection process must produce few, if any, false positives. A large number of false positives increases the overhead as more unnecessary removal processes will be applied. This may also increase the percentage of mistakenly removed legal links. In order to successfully remove all the wormhole links, the removal process must be used by all nodes located within the range of the wormhole – these are the nodes inside the two red circles in Figure 4.2.

4.2.1 The Detection Process

The detection process consists of three main operations: initial detection, mutual detection, and cooperation and control rules. The initial detection operation is basically a modified version of the DeWorm protocol, described in the previous chapter. In this case a node wants to check if a neighbor is "real" or reached through a wormhole or whether the node itself is in the vicinity of the wormhole. By "real", we mean that the neighbor is actually located within the node's transmission range. A neighbor may be a "real" neighbor, yet be in the vicinity of a wormhole transceiver. The detection process does not distinguish between such nodes and nodes that are reached through a wormhole. The removal process is responsible for that step. In this section the network model will be first described, then the initial detection operation, the mutual detection operation, and the co-operation and control rules will all be presented both briefly and with details.

4.2.1.1 Network Model and Notation We will start by describing the network and the attack model used. Consider an arbitrary ad hoc or sensor network consisting of n nodes represented by the ordered set Q. Let the set of one-hop neighbors of a node A be N_A, that is $N_A = \{A_1, A_2, ...A_{k_A}\}$, where k_A is the number of neighbor replies received by node A. The wormhole equipment $M_1 \leftrightarrow M_2$ is defined as two extra nodes M_1 and M_2 that are not part of the network, i.e., not elements of Q. Here we assume a closed wormhole where M_1 and M_2 are not visible to their neighbors (i.e., they do not advertise their node IDs or MAC addresses) and that the wormhole is an out-of-band physical layer wormhole that uses a high speed link to connect M_1 and M_2. Detecting such wormholes is considered to be extremely difficult [31]. The set of one-hop neighbors of M_1 and M_2 will be N_{M_1} and N_{M_2}, respectively. Note that by definition, every node in N_{M_1} is connected to all the nodes in N_{M_2} via the wormhole and vice versa. Thus N_A, the one-hop neighbor set of node A includes nodes both within transmission range and on the other side of the wormhole if A is in the transmission range of the wormhole. Let $\hat{N}_A$ be the set of "true" one hop neighbors of A. Then $N_A^* = N_A - \hat{N}_A$ will be the set of nodes that are not true neighbors of A. Clearly, $N_A^* = N_{M_2}$. With reference to Figure 4.10, $N_A = \{B, C, D, F\}$ and $\hat{N}_A = \{C, D\}$ and $N_A^* = \{F, B\}$ (Type 1 neighbors). The set $\hat{N}_A$ comprises of nodes that may also belong to N_{M_1} – these are called Type 2 neighbors (e.g., D) and nodes that are not in N_{M_1} like C that are called Type 3 neighbors. Let the route from any node X to any node Y be R_{X-Y} and $|R_{X-Y}|$ be the length of the route in number of hops. For the discussion in this section we will assume the existence of a single two-end wormhole.

4.2.1.2 Initial Detection Operation

Node A will first determine if it is in the vicinity (within the transmission range) of a wormhole. The process used here is similar to that described in Chapter 3. The basic idea here is as follows. If node A is in the vicinity of a wormhole, one or more nodes in N_A will be on the other side of the wormhole. Suppose that $B \in N_A^*$ is not a true neighbor of A and that the wormhole is η hops long. If a node $X \in \hat{N}_A$ were to find a route to some neighbor of B that is not a neighbor of A (called the *target* T) avoiding all nodes $\in N_A^*$, such a route must be at least η hops long (since a route that goes through the wormhole has to include some node in $N_A^* = N_{M_2}$, the wormhole is avoided and the alternate route must be at least as long as the wormhole itself). Implementing this idea is not trivial since node A does not know the composition of N_A^*. So node X avoids using all nodes in N_A which will include all nodes in N_A^*. But X itself may be part of N_A^* making it necessary for all nodes in N_A to repeat this process. Further, if $T \in N_{M_1}$, it could be closer to A than B. All of these are taken into account in the algorithm to detect existence of a wormhole shown in Figure 4.3. A detailed description of all the steps with some discussion is presented next.

Step(1): Node A will discover its one-hop neighbors by broadcasting a "hello" message. Cryptographic techniques (e.g., authentication) are used to prevent malicious nodes from

```
***Detection***$(A,B)$

1. $A$ **finds** $N_A$
2. $B$ **sends** $N_B$ to $A$
3. $A$ **chooses** $T \in \{N_B - N_A\}$
4. $\forall X \in N_A$, $X$ **finds** $R_{X-T} : N_B \cup N_A \not\subset R_{X-T}$ and $|R_{X-T}| > 1$
5. $\forall X \in N_A$, $X$ **sends** $|R_{X-T}|$ to $A$
6. $A$ **determines** *Select*$(R_{X-T})$ and **computes** $|R_{S-T}|$
7. **If** $|R_{S-T}| > \eta + 3$ **then** $A$ assumes $B$ is connected to it through a wormhole
```

Figure 4.3: The Initial Detection Algorithm

sending fake replies.

Step(2): Node A receives replies from its neighbors and verifies their authenticity. Neighbors could be elements of $\hat{N}_A$ or N^*_A

Step(3): Node A wishes to determine if B is a true neighbor. A asks B to provide its one-hop neighbor list N_B. We refer to B as the neighbor under examination.

Step(4): Node A picks some node $\in N_B - N_A$ and marks it as the target node T.

Step(5): Node A will ask all its one-hop neighbors (real and purported) to find the shortest route to T. Those routes must: (i) cannot be direct (must pass through another node) and (ii) avoid the one-hop neighbors of both A and B.

Step(6): Nodes in N_A reply to A with the length of their shortest routes to the target node T.

Step(7): Node A employs *Select(route)* (see Figure 4.4) to select a route that will be compared with the wormhole route (the wormhole route should be 3 hops – from the neighbor of A to A, from A to B and from B to T). Using *Select(route)* eliminates extremely long route outliers while ensuring that a route that is η hops longer than the wormhole route is not missed.

Step(8): If the difference between the length of the selected route and the wormhole route is greater than the wormhole length η then the neighbor under examination is connected by a wormhole link.

```
Select($R_{X-T}$)

1. $\forall X \in N_A - \{B\}$
   sort $|R_{X-T}|$ from longest to shortest
   ${R_{X-T}}^L$ is longest, ${R_{X-T}}^1$ is shortest
2. $R_{S-T} = {R_{X-T}}^L$
3. for($i = 1,\ i < L,\ i++$)
   { if$\left(\left|{R_{x-T}}^L - {R_{x-T}}^{L-i}\right| \leq \eta\right)$
   then $R_{S-T} = {R_{X-T}}^{L-i}$
   else break }
```

Figure 4.4: The Route Selection Algorithm

Selection of Route: The algorithm used by node A to find R_{S-T} is *Select*(R_{X-T}) and shown in Fig. 4.4. This is similar to the algorithm presented in Chapter 3. Node A creates a sorted list of route lengths from its neighbors to T (excluding replies from neighbors that do not have routes to the target node). Node A picks a route that is smaller than the longest route by *not more than* η if it exists. Otherwise the longest route is picked. It is the length of the picked route that is used to determine the existence of the wormhole. We have tested other methods for *Select*(R_{X-T}). Using the longest route has a better detection rate especially for short wormholes but increases the percentage of false positives for randomly distributed networks. Using the average length of all routes reduced false positives but also reduced the detection percentage. The method in Fig. 4.4 provides the best performance.

Selection of η: The value of η is not known a priori, but while implementing security in the network, the administrative entity can decide what it should be. Typically, longer the wormhole is, greater is the damage. With $\eta = 2$ even short wormholes (2 hops) can be detected. However, the simulation results will show that the number of false positives will be high. Using $\eta \geq 3$ reduces false positives but short wormholes (less than 3 hops) may escape detection. $\eta = 2$ or 3 provides the best tradeoff between detection rate and false

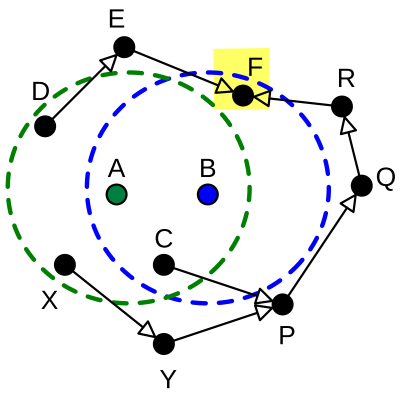

Figure 4.5: Detection Operation without Wormhole

positives.

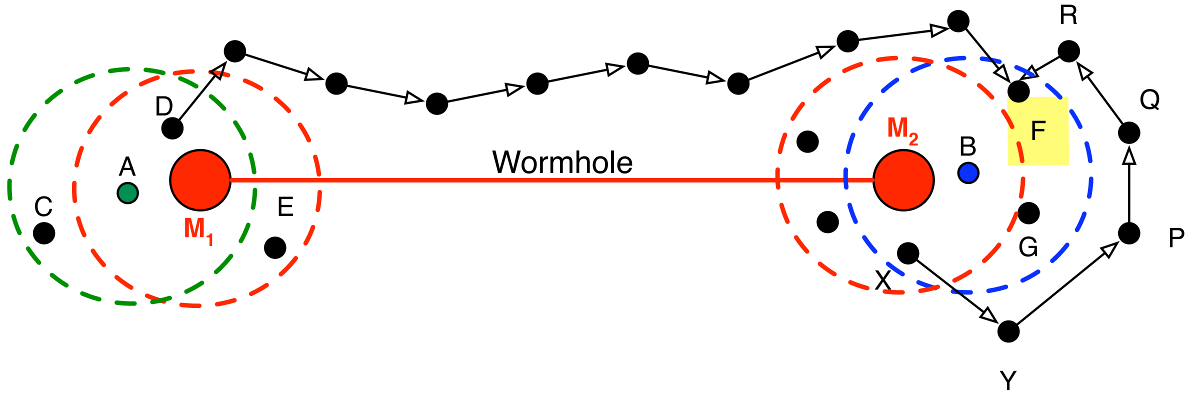

Figure 4.6: Detection Operation with Wormhole - case 1

No Wormhole: In the example in Fig. 4.5 nodes A and B are real neighbors – there is no wormhole in this case. Node A wants to check if node B is a real neighbor. Node A picks node F as the target node and asks its neighbors C, D, and X to find routes to node F. The lengths of these routes will be 4, 2, and 5 hops. Note that the nodes have to avoid the one hop neighbors of nodes A and B in their routes to F. Node A will select one of these routes (for now let us assume it is the longest one of 5 hops). The route from X to F through A will be 3 hops. If $5 - 3 < \eta$ then node A will decide that node B is a real neighbor. In some cases, $|R_{X-T}| - 3 \geq \eta$ if the topology is sparse and there is a false positive.

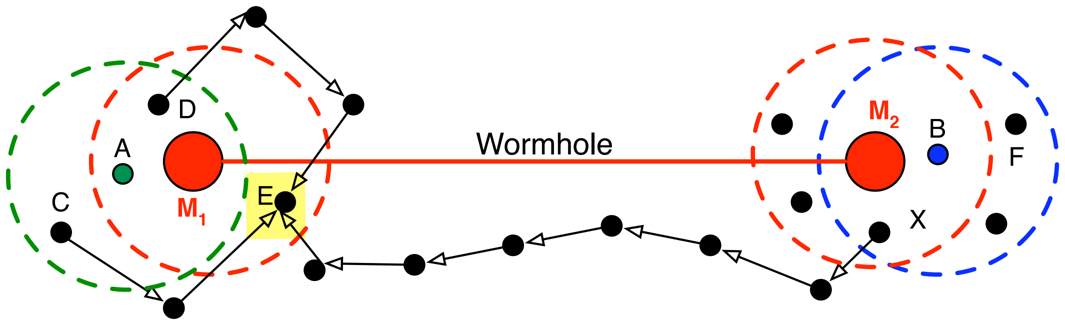

Figure 4.7: Detection Operation with Wormhole - case 2

With Wormhole: Cases where nodes A and B are connected with a wormhole are shown in Figs. 4.6 and 4.7. In Fig. 4.6, $A \in N_{M_1}$ and it has at least one neighbor $B \in N_{M_2}$. The target node must be a neighbor of B but not of A. Thus, there are two possibilities for the target node in this case. An example of the first is node $F \in \hat{N_B} - N_{M_2}$ in Fig. 4.6. Neighbors of A avoid other nodes in N_A and all nodes in N_B when they try to reach F. Since all nodes in N_{M_2} are included in N_A and all nodes in N_{M_1} are included in N_B, all the wormhole links will be avoided. True neighbors of A will have routes to F that are longer than 3 hops by at least η and the wormhole will be detected. For instance, node D, which is a true neighbor of node A cannot use nodes B or X to reach the target node F and will use the long route shown in Fig. 4.6. In the second case, the target node is an element of N_{M_1} but outside the range of A (e.g., node E in Fig. 4.7). In this case, true neighbors of A will find short routes to E, but purported neighbors $\in N_{M_2}$ will have long routes to E (e.g., node X is Fig. 4.7). To conclude, in either case, some neighbor of A will report a route whose length exceeds $3 + \eta$ and the wormhole is detected.

In the next section we present approaches to reduce false positives and to reduce the overall number of checks that must be performed between supposed neighbors in the network.

4.2.1.3 Mutual Detection Operation

False positives can occur in two ways - first when there is no wormhole and the topology is sparse resulting a long route to the target and second when node A tries to check for the existence of a wormhole between itself and a Type 3 neighbor (see Fig. 4.10). In the latter case, nodes $\in N_A^*$ will find long routes to a Type 3 neighbor. As already mentioned, at this point, simply using the detection scheme will not enable distinguishing between Type 1 and Type 2 neighbors.

We have found that mutual detection (i.e., A checking if B is a true neighbor and B checking if A is a true neighbor) reduces the percentage of false positives significantly. A wormhole will be suspected to exist **if and only if** both A and B discover their links to be connected through a wormhole. When there is no wormhole, the target nodes for nodes A and B (see Fig. 4.5) when they perform mutual checks will be different. In most cases, even if A marks node B as connected through a wormhole, B will not or vice versa. Of course there will still be topologies where both A and B will still flag each other as connected through a wormhole, but this fraction is an order of magnitude smaller as shown in Section IV. For instance, if $P_{f+ve} \leq 10\%$ is the probability of false positive to accur when the initial detection operation is used then using the mutual detection principal may reduce the probability of false positive to ${P_{f+ve}}^2$ that is $\leq 1\%$. Similarly, even if node A in Fig. 4.10 flags node C, node C will not flag node A as it has no neighbors $\in N_{M_2}$. Thus false positives are reduced and Type 3 neighbors correctly identified.

4.2.1.4 Co-operation and Control Rules

The question here is whether we need every node in the network to check all its neighbors or is it possible to exempt some nodes from applying the initial detection process. Having every node check all its neighbors will cause a large overhead and delay, especially with dense networks with high average node degree. We will start by discussing simple possible approaches that could be used by any secure neighbor discovery protocol to reduce the overhead and show why these approaches will not work for all cases. Then we will present a set of co-operation and control rules that make use of the special characteristics of the detection operation to significantly reduce the overhead and improve the protocol scalability.

A simple approach to reduce overhead could be to exempt one hop neighbors of each node that used the initial detection operation from checking their neighbors if no wormhole was detected. This approach could work if the wormhole is connecting more than one node from one side of the network with node(s) at the other side, similar to the example shown

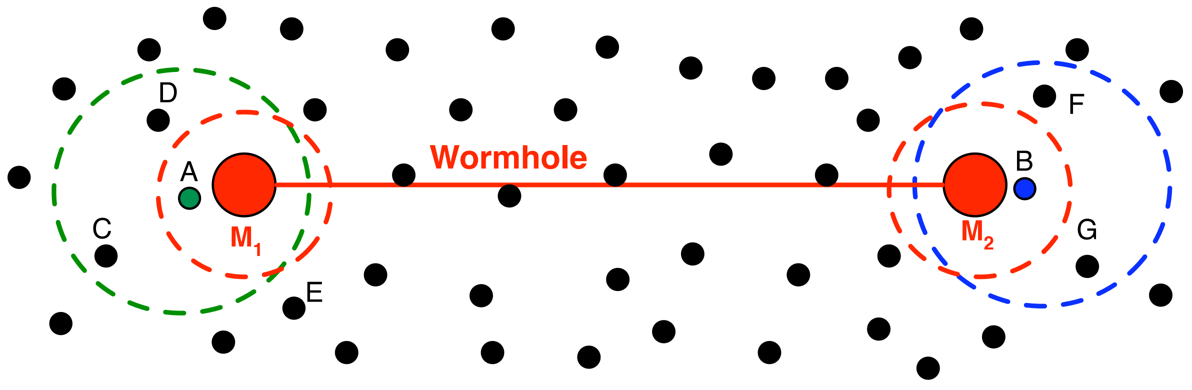

Figure 4.8: Wormhole Connecting Single Nodes

in Figure 4.2. In such a case case, the probability of not detecting the existence of the wormhole is very small. On the contrary, if the wormhole is connecting exactly one node with another node located several hops away, as in the example shown in Figure 4.8, then this approach will not guarantee the detection of the wormhole. This is because the nodes that are connected using the wormhole – nodes A and B – may now be exempt from using the detection process, as one of their neighbors that is not within the wormhole range could have already used the detection operation. In our example, Figure 4.8, the wormhole will only be detected if either node A or B used the initial detection operation. Note that, their one hop neighbors and other nodes will be indirectly using the wormhole and are unable to detect the existence of the wormhole.

Another question is whether each node needs to check *all* of its neighbors or it can exclude checking some of its one hop neighbors. An approach that will reduce the number of nodes that need to check their neighbors is to exempt a node that is checked by any of its one-hop neighbors from being checked by its other one-hop neighbors. Let us suppose that a node is checked by one of its one-hop neighbors that is located within its transmission range but not within the transmission range of any of the wormhole transceivers (Type 3 neighbor). Then this neighbor will not detect the existence of the wormhole (which is true, as no wormhole is connecting it with the node). In this case, if this node was the first to apply the initial detection then none of the one hop neighbors that are located within the wormhole transceivers' range will check this node and thus the wormhole will not be detected.

More specifically, this approach will not successfully detect the wormhole that is shown in Figure 4.8 because of the mutual detection operation. In the example shown in Figure 4.8 if node B is checked by node F, which is one of its one hop neighbors, then node A will not check for a wormhole linking it to node B. Similarly, if node D checked node A for a wormhole link, then node B will not check node A, hence the wormhole will not be detected. Note that node A will detect something when it uses the initial detection operation to check node D. This is because when it asks node B to find a route to the target node, then node B cannot use node A in its route and will have a long route to reach node D. However, node D will not detect anything when it checks node A, because all its neighbors are on the same side as node A. Hence, the mutual detection condition operation will not be satisfied and the wormhole will not be detected.

In summary, neither one of the previous two approaches will guarantee detection for all cases. We have only shown one case where these approaches do not work. However there are other cases for which the approaches will not work as well.

To overcome all the issues that were discussed in this section we present simple rules that will ensure the detection of the wormhole in most cases. The rules should still be localized, which means they should only use information observed by the node or one of its one-hop neighbors. Without any formal proof, we argue that the following are simple rules for checking for existence of wormholes that eliminate a large number of unnecessary checks as verified by simulations.

1. If node A checks its link with node B and no wormhole is discovered, node B need not check its link with node A.
2. If node A checks its link with node B and a wormhole is discovered, node B must check its link with node A. This ensures that false positives are reduced as discussed previously.
3. If node A has checked its link to node B and vice versa, no wormhole is discovered, and any node C is a neighbor of *both* A and B, nodes A and B need not check their links with node C, and vice-versa.

A node that is involved in a mutual detection may expect any or all of the following:

- The node will also be involved in some mutual detection with other one hop neighbors.
- Some of the one-hop neighbors of the node will also either detect or be involved in a mutual detection with the same neighbor and/or other one-hop neighbors.

These observations from the one-hop neighbors are useful to confirm the nodes' detection decision, especially if the wormhole is connecting two or more nodes with two or more nodes.

However, because of the possibility of the special case shown in Figure 4.8, the the absence of any detection from the one-hop neighbors of any node will not rule out the mutual detection incidence. According to the above rules, node A in Figure 4.8 does not have a neighbor that is common to node B. Also node B does not have a neighbor that is common to node A. This implies that, according to the rules of the co-operation and control operation, either node A or B will have to check one another and hence the wormhole will be detected.

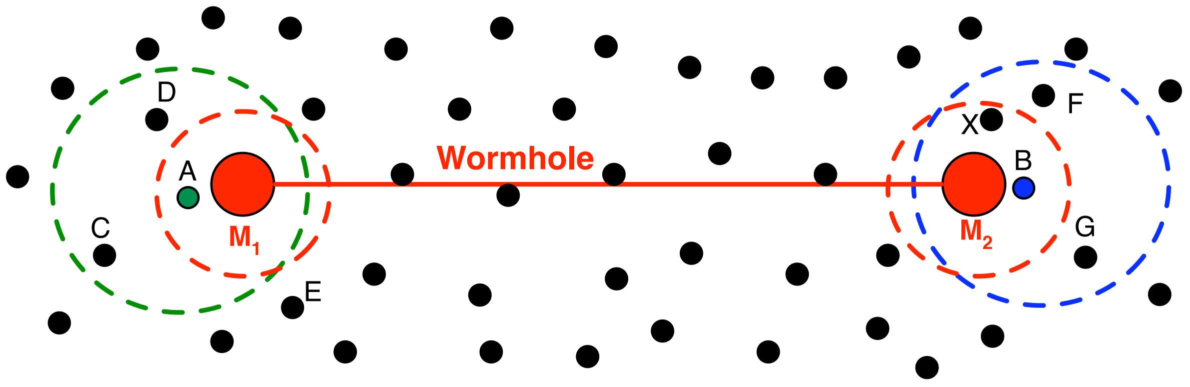

Figure 4.9: Wormhole Connecting Single Node with Two nodes

However, one may expect the situation shown in Fig. 4.9 to be problematic, where node A has two neighbors X and $B \in N_A^*$. What if X checks its link with B and finds no wormhole? One of the characteristics of the detection process is that it cannot distinguish between Type 1 and Type 2 neighbors. Since both A and B are neighbors of X, when X checks its link with B, it will ask A to find a route to a target node (say G) and this will reveal the presence of the wormhole.

Applying the rules described in this section will significantly reduce the number of initial detection operations that must be performed by nodes to their one-hop neighbors. Obviously many nodes will not need to apply the detection operation as all their neighbors may have been checked by other common neighbors. Later our simulations and analysis will show that the number of initial detection processes can be reduced by up to 80% compared to the case where all the nodes needs to check all their one-hop neighbors. This will also drastically reduce the overhead caused by the overall network detection process. If nodes in the network are having different energy level then the co-operation and control operation could be enhanced to increase the network life time. This could be acheived by making the nodes with higher evergy level check their neighbors before others. This will reduce or eleminate the involvment of low energy nodes in the detection operation as to save their energy.

4.2.2 Removal Process

Detecting the existence of wormholes in the network is an important step. However, another crucial process is to remove the links created by the wormhole. Note that a wormhole that connects m_1 nodes $\in N_{M_1}$ with m_2 nodes $\in N_{M_2}$ results in $2m_1m_2$ bogus links. The removal process is not simple as the failure to remove a single link almost means the failure of the removal process. This is because the existence of such link means the existence of a wormhole which can still cause the same damage to the network. Even if one of these links is not removed it will still cause damage by attracting traffic. Many of the available wormhole defense mechanisms ignore the removal of the wormhole connected links or use techniques that may remove many legal links.

Fortunately, the detection process, presented earlier in this chapter, is extremely effective when it comes to detecting a wormhole that actually exists. As previously described, the detection process flags the existence of a wormhole. A link between both Type 1 neighbors and Type 2 neighbors will be flagged as corrupted by a wormhole (and this is confirmed by mutual checks). However, mutual checks between Type 3 neighbors allows them to identify the fact that they are not connected through a wormhole. The challenge then is to distinguish between Type 1 and Type 2 neighbors to avoid removing legal links between Type 2 neighbors. We recap the definitions of types of neighbors that are shown in Figure 4.10:

Type 1: Neighbors, as nodes B and F, those are located on the other side of the wormhole, $\in NB_A$ and $\in NB_{M_2}$.

Type 2: Neighbors, as node D, those are located within the nodes' transmission range and also within the wormhole range, $\in NB_A$ and $\in NB_{M_1}$.

Type 3: Neighbors, as node C, those are located within the nodes' transmission range but not within the wormhole range, $\in NB_A$ and $\notin NB_{M_1}$ and $\notin NB_{M_2}$.

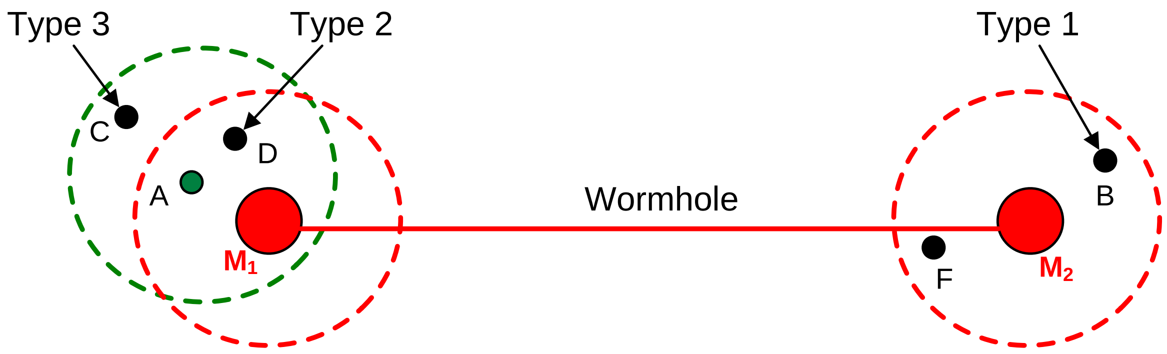

Figure 4.10: Types of Neighbors

Links with Type 1 neighbors will be detected with the detection process and will be mutually confirmed. Hence they are expected to be completely removed. When node A wants to detect a neighbor of Type 1 then the neighbors of node A (e.g., node C in Figure 4.10), will have have long routes to the target node. Links with Type 2 neighbors will also be mistakenly detected with the detection process and will be mutually confirmed as being corrupted by a wormhole. If the detection process was used to remove links, then links to Type 2 neighbors will all be mistakenly removed. Links with Type 3 neighbors may be flagged as bogus with the initial detection process but will not be mutually confirmed. Thus they are not expected to be mistakenly removed.

The reason why links with Type 2 neighbors are detected is because both nodes will have neighbors on the far side of the network. In the example in Figure 4.10 nodes A and

D will both have nodes within M_2's range as their neighbors (e.g., nodes B and F). Thus routes from those neighbors (nodes B and F) that avoid other one-hop neighbors will be long enough to trigger the detection of the wormhole. On the other hand, neighbors of Type 3 will not have neighbors located on the far side of the network. Hence, the mutual detection operation will not confirm the existence of a wormhole. This is because Type 3 neighbors will not detect node A with the initial detection operation. In the example, node C does not have neighbors on M_2's side (such as nodes B and F).

In conclusion, the main challenge of the removal process is to distinguish between neighbors of Type 1 and Type 2 in a way that will always ensure the removal of all links to neighbors of Type 1 and remove very few, if any, of the links to neighbors of Type 2. Given the fact that both neighbors of Type 1 and 2 will be flagged and confirmed by the detection process, this is crucial.

The removal process need only to be applied by all nodes that are within the range of any of the wormhole transceivers (M_1 and M_2 in the case of single two-ended wormhole). As previously explained, the detection process will almost always detect the wormhole. Thus, all the one-hop neighbors of both nodes involved in a detection process that detected a wormhole as existing in the vicinity, will be required to apply the removal process to links with all of their neighbors. The main reason behind that is to ensure that all nodes that are located within the range of both ends of the wormhole have applied the removal process and thus all the bogus links created by the wormhole will be removed. In the example shown in Figure 4.2 nodes A and B will mutually detect each other. The neighbors of node A will include all nodes in the range of M_2, i.e., nodes X, F, and B. Similarly, neighbors of B will include all nodes within the range of M_1, i.e., nodes A, D, and E. Thus, having all neighbors of nodes A and B in Figure 4.2 be involved in the removal process will ensure that all links created by the wormhole are checked. This means that links with neighbors of Type 1 and 2 are all checked with the removal process.

The removal process consists of two main operations: the initial removal operation and a mutual removal operation. The initial removal operation will check the links to decide whether they are of Type 1 or 2 and the mutual removal will be used to confirm this result.

4.2.2.1 Initial Removal Operation The initial removal operation will be used by all nodes located within the range of the wormhole to check the links with all of their neighbors. Let us assume that node A used the detection process and detected a wormhole as existing when it checks the link with node B. Node A wants to use the initial removal operation to check if node B is a Type 1 neighbor and decide if it has to remove the link to node B. The initial removal operation works as follows:

- Node A will only ask all its one-hop neighbors that are not also neighbors of node B to find the shortest "non-direct" routes to one of node B's neighbors that is not a neighbor of A. In other words, node A will ask all its neighbors that are not part of N_B to find routes to neighbors in N_B that are not part of N_A *one-by-one*. Again, the routes must not pass through any of the one hop neighbors of nodes A or B, which are nodes in N_A and N_B.
- Node A will use the route selection mechanism (Figure 4.4) to determine the length of the selected route that will be compared with the direct route.
- If the difference between the selected route and the direct route is greater than η then node B will be assumed to be Type 1 neighbor and link from A to B will be removed.
- Otherwise (If the difference between the selected route and the direct route is NOT greater than η) the next neighbor of node B that is not a neighbor of A will be checked and so on.
- If all the neighbors of node B that are not neighbors of A were checked and none of them reveals the existence of a wormhole, then node B will be assumed to be of Type 2 and the link from node A to B will not be removed.

The removal algorithm that should be used by node A to decide the removal of the link to node B is shown in Fig. 4.11. The question is, how does this removal process successfully discriminate between the neighbors of Type 1 and (2)? Consider a Type 1 neighbor (see Fig. 4.12) B of node A. Node A can have neighbors in $N_A - N_B$ that are on either side of the wormhole as also node B. This is the reason why nodes in $N_A - N_B$ should find routes to nodes in $N_B - N_A$ one-by-one. Eventually, a long route is discovered. For example, node $H \in \hat{N}_A$ will find a long route to $K \in \hat{N}_B$ or node $X \in N_A^*$ will find a long route to $E \in N_B^*$.

```
Removal(A,B)

1. $\forall X \in N_A - N_B$,
   For some $Y \in N_B - N_A$,
   $X$ finds $R_{X-Y}$ : no element in $N_A \cup N_B$ is part of $R_{X-Y}$
   then $X$ sends $|R_{X-Y}|$ to $A$
2. $A$ uses all $R_{X-Y}$s and employs Select($R_{X-Y}$) to find $|R_{S-Y}|$
3. if $|R_{S-Y}| > \eta + 3$
   then $A$ removes link to $B$ and the process will stop
   else all $X$ will repeat process for next $Y \in N_B - N_A$
4. if no $|R_{S-Y}| > \eta$ $\forall Y \in N_B - N_A$,
   then link $A$ to $B$ will not be removed
```

Figure 4.11: Wormhole Removal Algorithm

It is not sufficient to pick any one node randomly in $N_B - N_A$ as it is possible that $N_A - N_B$ has nodes on only one side or the other of the wormhole. In the case of a Type 2 neighbor B (see Fig. 4.13), nodes in $N_B - N_A$ and $N_A - N_B$ are *both* constrained to be on the same side as nodes A and B.

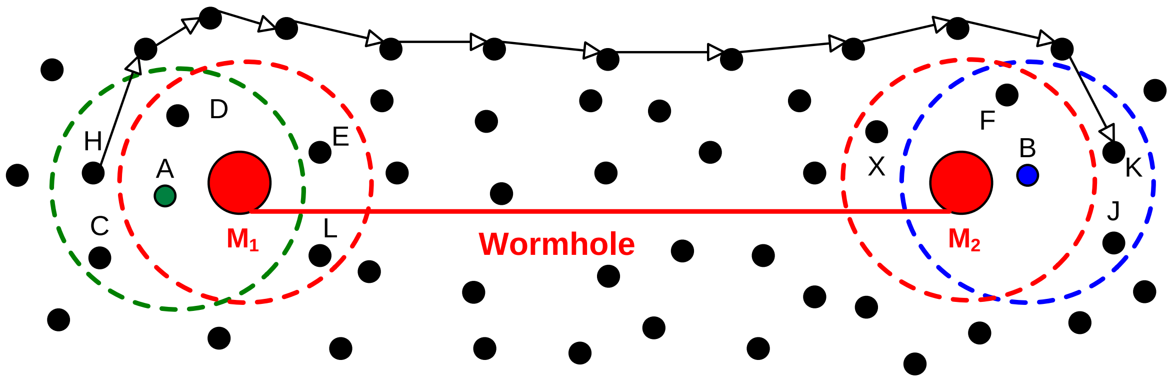

Figure 4.12: Removal of Type 1

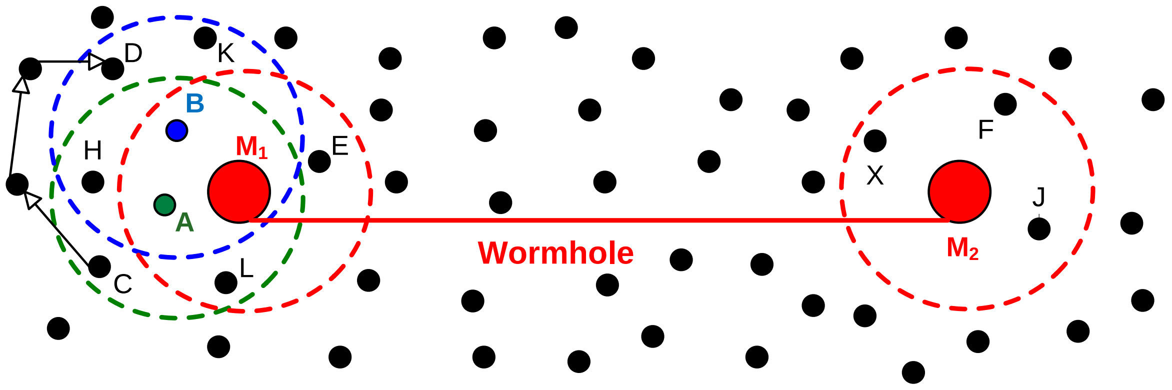

Figure 4.13: Removal of Type 2

Let us now show how will a link to Type 2 neighbor not be removed by mistake. Any neighbor of A that belongs to N_A^* will also belong to N_B^*. Thus, routes from nodes in $N_A - N_B$ to nodes in $N_B - N_A$ will likely be short (e.g., from node C to node D in Fig. 4.13). In the example shown in Figure 4.13 node A wants to check if the link to node B is bogus or not (B is a Type 2 neighbor of A). If a node has a neighbor of Type 2 then all the non-common neighbors of the two nodes will be located on the same side of network. This is because the neighbors of both nodes that are located at the other side of the network (connected by the wormhole) will all be common neighbors. In the example in Figure 4.13, node A has node B as Type 2 neighbor and since both nodes are within M_1's range, all nodes in M_2's range will be common neighbors. That is, nodes X, F, and J are all common neighbors to both A and B. Thus, none of these nodes will be asked by node A to reach the target node. Moreover, none of these nodes will assigned as a target node because they are all common to both nodes A and B. More specifically, all the neighbors of A that are not common neighbors of B, i.e., nodes C and L, that need to find routes to the target node, and the neighbors of B that are not common to A, nodes D and K, that will be target nodes, are all located at the same side (as M_1 in this example). Consequently, the routes are not expected to be long, similar to the one from C to D shown in Figure 4.13, and the link between A and B (Type 2 in this case) will not be mistakenly removed.

4.2.2.2 Mutual Removal Operation In a manner similar to the mutual detection process, the accuracy of the removal process can also be improved using mutual removal. Since links to nodes located at the other side of the wormhole will always be removed, a node can verify the correctness of its decision to remove a link. Each node that removed a link will contact the node to which the link was removed and if the other node also has decided to remove the link to the node then the node will verify the accuracy of its removal. Otherwise, if the other node is not removing the link then the node may decide not to remove the link.

4.3 DISCUSSION

In this section, we reiterate some aspects that were previously mentioned in more detail.

4.3.1 The Wormhole Length Parameter

Wormholes are at least longer than the transmission range of a node (otherwise their impact is minimal). Similar to the sensitivity parameter discussed in Chapter 3, η is a parameter that determines what wormhole lengths the protocol is capable of detecting. With $\eta = 2$ even short wormholes (2 hops) can be detected. However, the simulation results will show that the number of false positives will be higher for $\eta = 2$. Using $\eta \geq 3$ reduces false positives but extremely short wormholes (less than 3 hops) may escape detection. Thus, $\eta = 2$ or 3 provides the best tradeoff between the detection rate and false positives.

4.3.2 Route Selected for Comparison

To determine the selected route from the set of routes found to the target node T for comparison with η to detect the wormhole we used a method that provides the best detection and a reasonable number of false positives. This is similar to the method used with DeWorm in Chapter 2. The algorithm used by node A to find R_{S-T} is $\mathit{Select}(R_{X-T})$ and shown in Fig. 4.4. The node creates a list from the replies containing route lengths to the target node from its neighbors and sorts them according to their lengths from longest to shortest (excluding replies from neighbors that do not have routes to the target node). The node picks the length of a route that is smaller than the longest route by *not more than* η if it exists. Otherwise, the length of the longest route is picked.

In section 3.9.3.1 we have tested other methods that could be used to pick the selected route. Using the longest route gives a better detection rate than the one we are using here, specially for short wormholes. However, the percentage of false positives was much higher, especially for randomly distributed networks. A method that used the average of all routes yields a lower percentage of false positives but also its detection percentage was lower. The method that is used ($Select(R_{X-T})$) provides the best performance overall.

4.3.3 Limitations

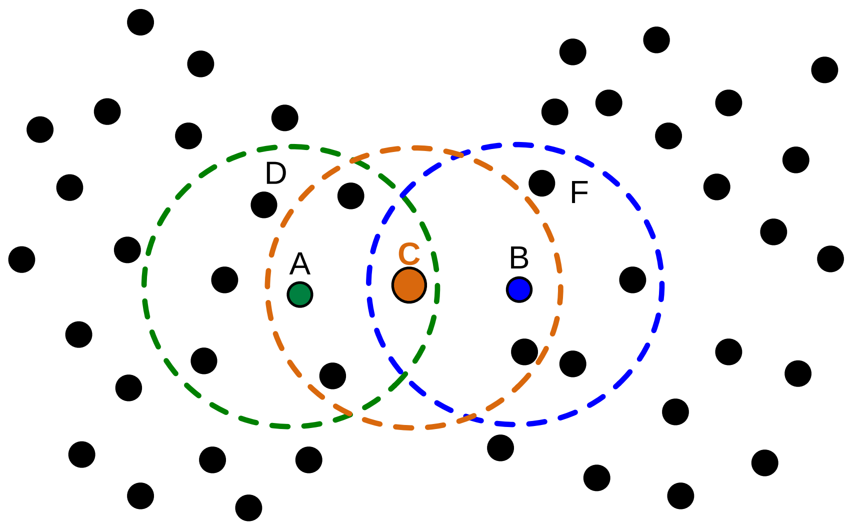

Figure 4.14: Critical Node Example

If a *critical node* (if this node is removed, the network will be partitioned) exists in the the network then some of its neighbors will not be able to find any route (to a target node) that avoids the critical node. Figure 4.14 shows an example of a critical node in a network. Node C is a critical node and if removed, nodes from the left side of the network will not be able to communicate with nodes on the right side of the network. In this case node A wants to check node C and used node F as the target node. None of node A's neighbors (for example node D) will be able to reach node F without using node C. Node A will either assume that there is a critical node or a wormhole is trying to connect two isolated networks. If critical nodes are few and do not change in the network, one possible solution to this issue is to use a protocol that can identify critical nodes [53, 54] a priori that will cooperate with our protocol. A critical node will then be exempt from using the initial detection operation to check its neighbors and also will not be checked by its neighbors.

4.3.4 Impact of mistakenly removed links

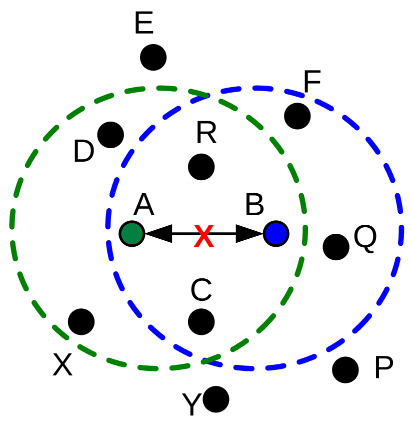

Figure 4.15: Removing Legal Link

While the number of legal links removed will be small (as shown in Section IV), the impact of removing a very small number of legal links can be expected to be minimal. For example, in Fig. 4.15(b), if the link from A to B is removed, node A may be able to use node C to reach Q without an increase in the number of hops. In some cases, a few additional hops may be required. Finally, we have not explicitly described protocols that will use *Removal*() (and employ mutual checks and exchange of information). We can expect this to operate in a manner similar to the detection process.

4.3.5 Other Issues

It is worth to discuss the differences between the initial detection and the initial removal operations. Why do we not apply the the initial detection to decide the removal of a link? The answer to this question, as already discussed, is that the initial detection will always give false positives when used to detect Type 2 neighbors. Thus using the initial detection process to remove links may lead to isolating the nodes that are within the range of the wormhole, specially if the transmission range of M_1 or M_2 is large.

Moreover, why do we not to use the initial removal operation for initial detection? In fact, we have thought about this idea and tested it using simulations. We observed that it leads to significantly large number of false positives (when no wormholes exist) - something in the range of 35% compared to 1% using the combination of initial and mutual detection processes. Let us have a closer look at the two operations to undestand why we observe this large difference in false positives. In the case where nodes A and B are real neighbors, with the initial removal process node A will only ask its neighbors that are not common to node B to find routes to the target node. Obviously, these routes will be the longest compared to the routes from the other neighbors of node A. In networks where the nodes are randomly distributed there is a higher probabilty to have a long route that may mistakenly trigger the removal prcess causing false positives. Besides, with the removal operation we are not only picking randomly one of node B neighbors that is not common with node A to be the target node, but we will make all of them target nodes one after another and decide that it is a wormhole if any of them is detected by the removal operation. Futhermore, this also has to do with the way the selected route algorithm works, where if we have more routes that are decreasing in length, which is the case with the initial detection operation, it is more likely to choose a shorter selected route and hence less likely to cause a false positive.

Another case that is less likely to occur is the situation when node A does not have any neighbors on M_2's side and the target node is picked on M_1's side. In the example shown in Figure 4.7, if node X is not there and only node B is connected to node A via the wormhole then all neighbors of A will find short routes to the target node E. Thus, the detection operation as is will fail to detect the wormhole. It is less likely for this situation to happen, as node B usually has more neighbors that are non-common with node A and located at M_2 side, hence it is more likely to pick one of these nodes as target node. Moreover, wormholes transceivers normally have equal ranges. Thus, if the range of M_1 is large to include possible target nodes then the range of M_2 will also be large to include at least one node other than node B.

To overcome this problem the detection operation can slightly modified. Instead of picking a single target node randomly, node A will ask its neighbors to find routes to all possible target nodes. Obviously node B must have at least one neighbor that is located on M_2's side and the wormhole will be detected.

4.4 PERFORMANCE EVALUATION

4.4.1 Simulations

The important performance metrics for secure neighbor creation are: the percentage of correct detection of the wormhole, the percentage of false positives, percentage of wormhole links removed, and percentage of legal links removed by mistake. We have considered two different node distribution models: grid distribution with some perturbations and random distribution. For the random node distribution, the coordinates of the nodes (x_i, y_i) for $i = 1, 2, ...200$ were independently and randomly chosen in the range from 100m to 2000m using a uniform [100-2000] random number generator. In the grid case, nodes are located in a perturbed 20×20 grid. The coordinates of each node x_i and y_j were randomly chosen using uniform random variables in the ranges $(100i - p100, 100i + p100)$ and $(100j - p100, 100j + p100)$, respectively, where p is the perturbation parameter and $i = 1, ...20$ and $j = 1, ...20$ (in our simulations, $p = 0.2$). After the nodes are distributed, the connectivity models UDG or Quasi-UDG and the transmission range determine the network topology. As in [63], we also investigated SECUND with two connectivity models the commonly employed unit disk graph and the quasi unit disk graph. The quasi-UDG connectivity model is available in [52]. Only results with the UDG are shown here for brevity. The results with the quasi-UDG connectivity model are very similar. To change the average node degree, the transmission range of the nodes was varied from 110m to 160m. The simulations were programmed in C using DSR routing protocol and node distribution models from ns-2. For statistical validation the simulations were repeated 50 to 100 times with confidence intervals of 95%. The secure neighborhood creation protocol was evaluated for networks without any wormhole, with two-ended wormholes, and multi-ended wormholes. ***Two-ended regular wormholes***: The two-ended wormhole connects groups of nodes from one location in the network to another group of nodes located several hops away (wormholes with different lengths are tested). As shown in Fig. 4.16, two separate wormholes are created in the network such that the ranges of the wormhole transceivers do not overlap. That is, each node in M_1's (M_3's) range is only connected to every node in M_2's (M_4's) area and vice versa. Let m_i be the number of nodes in the range of wormhole transceiver M_i. The number of links created by wormhole $M_1 \leftrightarrow M_2$ is $2m_1m_2$. Note that two nodes $A \in N_{M_1}$ and $B \in N_{M_2}$ have two links between between them $A \rightarrow B$ and $B \rightarrow A$. For the two wormholes shown in Fig. 4.16, the number of bogus links created is $(2 \times 4 \times 3) + (2 \times 4 \times 3) = 48$.

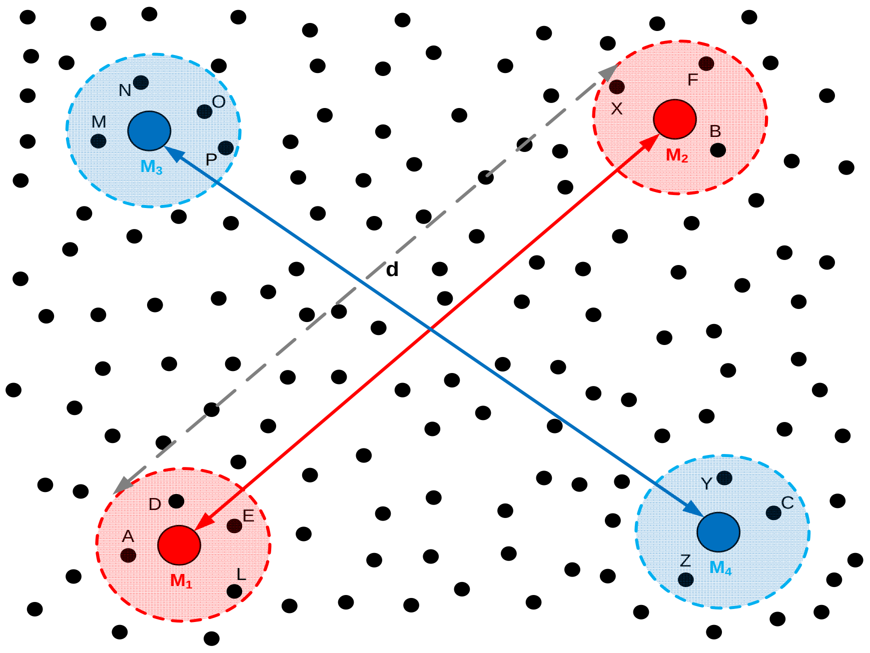

Figure 4.16: Two-ended Regular Wormhole

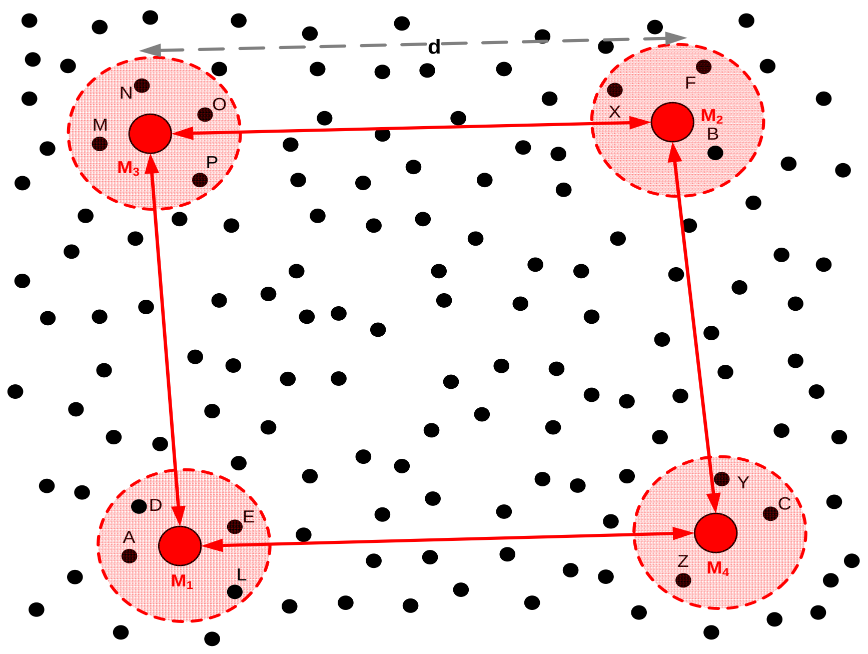

Figure 4.17: Multi-End Wormhole

Multi-Ended wormhole: In this case the wormhole will be connecting nodes located in many different areas. In the example shown in Fig. 4.17, each node located near any wormhole transceiver will be connected to all nodes located at the other transceivers. For instance, every node in N_{M_1} will be connected to every node in N_{M_2}, N_{M_3}, and N_{M_4}. In general, for a n-ended wormhole the number of links created will be: $\sum_{i=1}^{n-1}\left[m_i \cdot \left(\sum_{j=1, j\neq i}^{n-1} m_{j+1}\right)\right]$. For the example in Fig. 4.17 the number of links created by the 4-ended wormhole is $4\,(3+4+3)+3\,(4+4+3)+4\,(4+3+3)+3\,(4+3+4)=146$.

4.4.2 Results

4.4.2.1 False Positives and Mistakenly Removed Links

We first simulate networks without wormholes and run the *Detect*() and *Removal*() algorithms to determine the percentage of false positives and legal links removed by mistake.

In this case the network is simulated without a wormhole. Thus, any wormhole detected

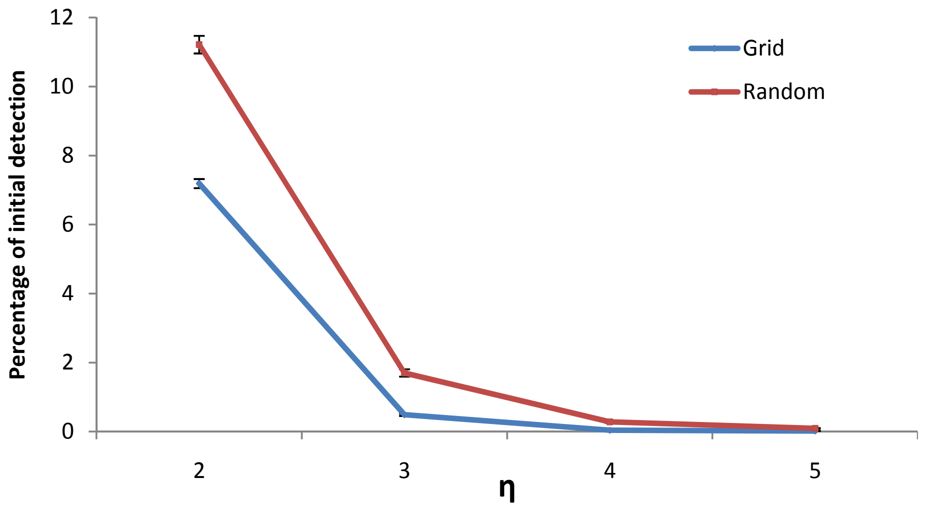

Figure 4.18: False positive with initial detection

by the initial detection operation will represent a false positive caused by this operation. The number of times the initial detection process will detect a wormhole divided by the number of times the initial detection process were used will indicate the percentage of false positives caused by the initial detection process. This result is shown in Figure 4.18 for both grid and randomly distributed networks and with different values of η. The result show that for both grid and randomly distributed networks the false positive rate decreases for higher η. With $\eta = 3$ the percentage of false positives is less than 2% and it almost reaches zero with $\eta \geq 4$. Note that with $\eta = 2$ the false positive rate will reach a high value, especially with randomly distributed networks. This is because it is possible for some nodes to find routes that are longer than the direct routes by more than 2 hops.

Figure 4.19 is similar to Figure 4.18 but with mutual detection operation used in this case (in fact, this represents the actual false positives produced by the detection process which includes mutual detection unless otherwise mentioned from now onwards). This value will have an impact on the removal process. As shown here, the percentage of false positives is significantly reduced – almost 0% for the Grid network and less than 0.2% for the Random network with $\eta = 3$. The reason why slightly more false positives occur with randomly

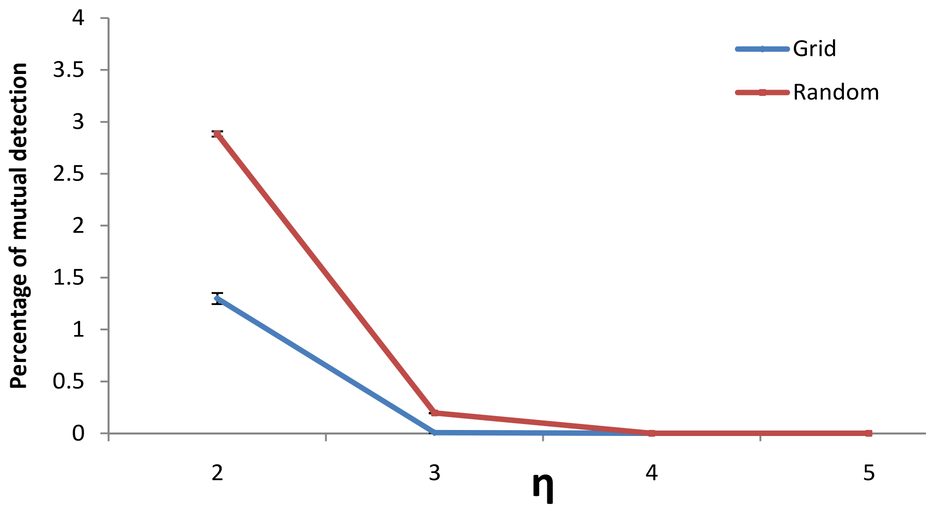

Figure 4.19: False positive with mutual detection detection

distributed network is due to the distribution of the nodes. Thus the nodes may have relatively long routes, compared to the direct route, to reach the target node.

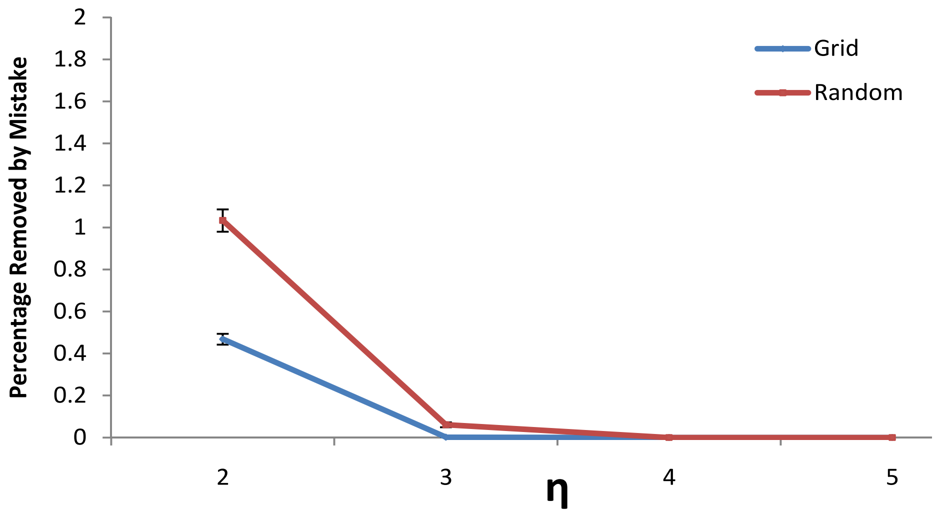

Figure 4.20: Percentage of legal links removed by mistake

The percentage of legal links removed by mistake is a very important parameter for secure neighbor discovery protocols. Figure 4.20 shows the percentage of legal links removed by mistake for both grid and randomly distributed networks with different values of η. The percentage in this case is computed by comparing the number of links removed to the total number of links available in the network. Note that the network in this case is simulated without a wormhole. The results show that our wormhole removal process removes none or very few legal links, this is due to the fact that in this case there are no Type 1 or Type 2 neighbors, discussed earlier in this chapter, which makes the process less complicated. For example, with $\eta = 3$ no links are removed by mistake with the grid network and only 1 or 2 links on average are removed by mistake from the entire simulated network.

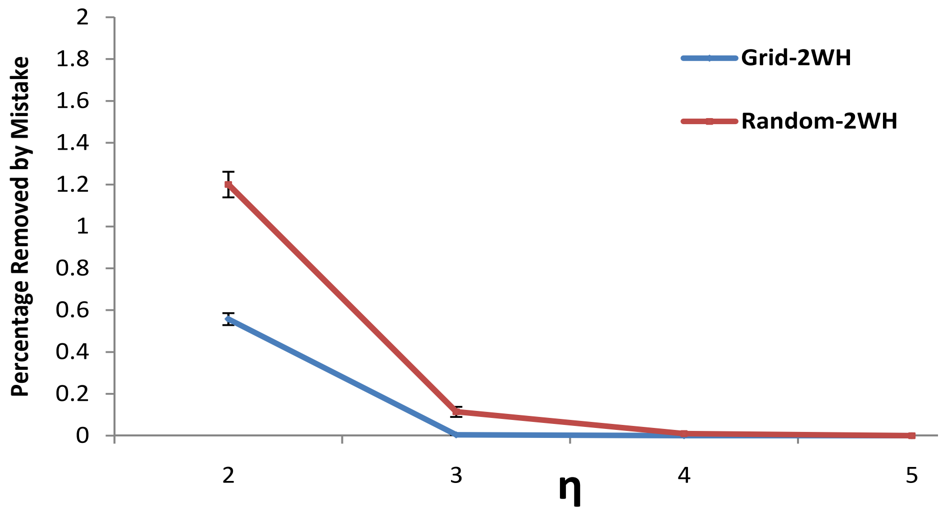

Figure 4.21: Percentage of links removed by mistake with 2 wormholes

4.4.2.2 Wormhole Removal The network is simulated with the two types of wormholes described earlier, two 2-ended wormholes and a 4-ended wormhole. Figure 4.21 shows the percentage of legal links removed by mistake with the existence of 2 two-end wormholes for both grid and randomly distributed networks. As expected, and due to the existence of the three types of neighbors, in this case the percentage of legal links removed by mistake is slightly increased compared to the network simulated without a wormhole, Figure 4.20. The percentage is still very low with $\eta = 3$.

Figure 4.22 is similar to Figure 4.21 but with 4-end wormhole simulated in this case. For the multi-end wormhole with $\eta = 2$ or 3 the percentage of legal links removed by mistake is

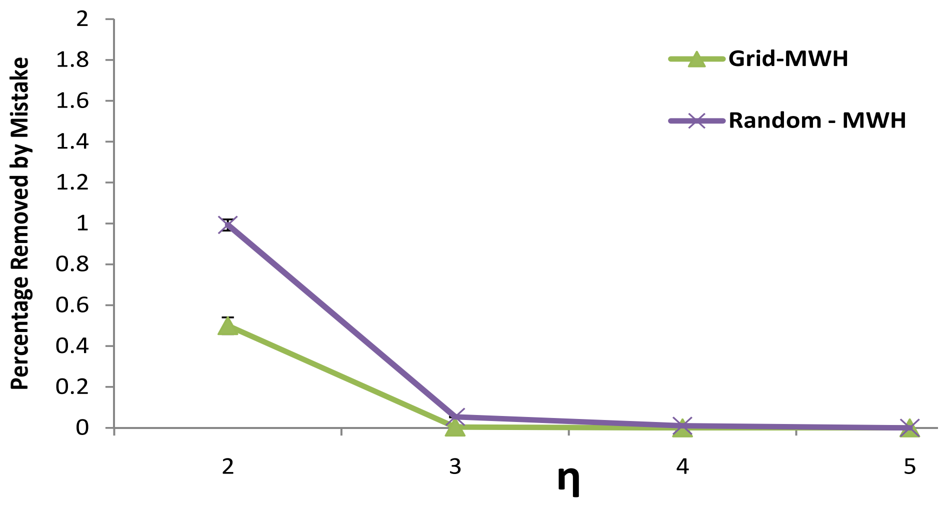

Figure 4.22: Percentage of links removed by mistake with multi-end wormhole

slightly less that the case with two-end wormholes. This could be due to the fact that the number of Type 1 neighbors is much higher for each node located within the range of the wormholes. Thus, it is easier to distinguish between Type 2 and Type 1 neighbors and fewer false positives will occur.

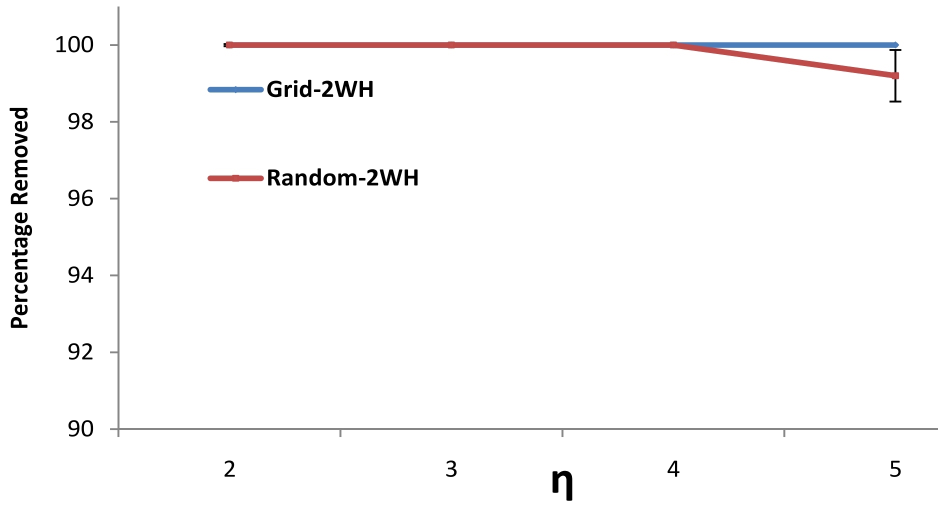

Figure 4.23: Percentage of wormhole links removed with 2 wormholes

Figure 4.23 shows the percentage of bogus links removed using the removal process for a network with two 2-end wormholes. The removal process will remove almost 100% of the wormhole links with $\eta = 2$ or 3 for both grid and randomly distributed networks. Figure 4.24 is similar to Figure 4.23 but with the multi-end wormhole considered in this case. In order to isolate the impact of the length of the wormhole for these two cases, the length of the wormhole was made > 5 hops. With $\eta \leq 4$ not all wormhole links will be removed, especially with the multi-end wormhole. This is because the length of the wormhole is not long enougth to distiguish between the wormhole links and legal links. Wormholes with short lengths are simulated in the next section.

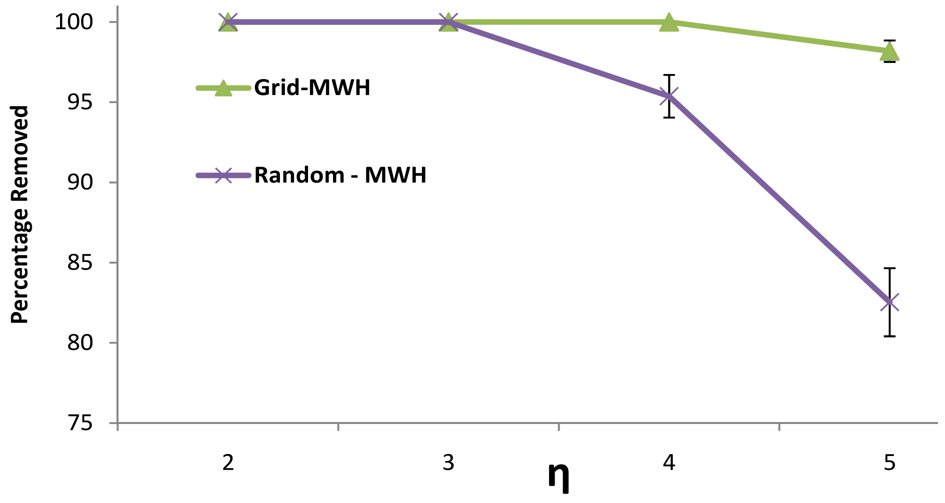

Figure 4.24: Percentage of wormhole links removed with multi-end wormhole

4.4.2.3 Impact of Wormhole Length

The impact of the length of the wormhole on the detection process is shown in Figure 4.25. Figure 4.25 shows the percentage of wormholes detected for different values of the wormhole length starting from 1.5 hops till 6 hops, and for three values of η: 1, 2, and 3. The results show that the detection process can still achieve 100% detection for very short wormholes. With $\eta = 1$ any wormhole can be detected, but the number of false positives will be extremly high. With $\eta = 3$ any wormhole with length that is ≥ 4 hops will be most certainly detected.

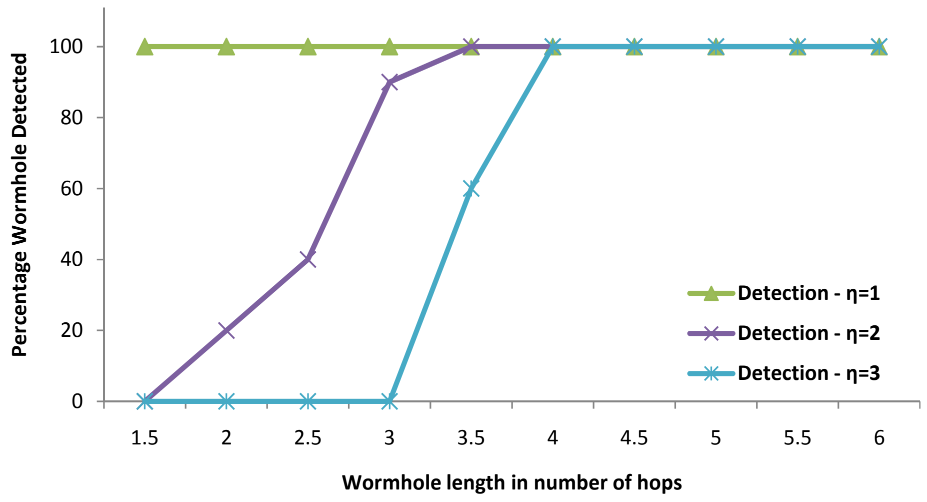

Figure 4.25: Impact of wormhole length on Wormhole Detection

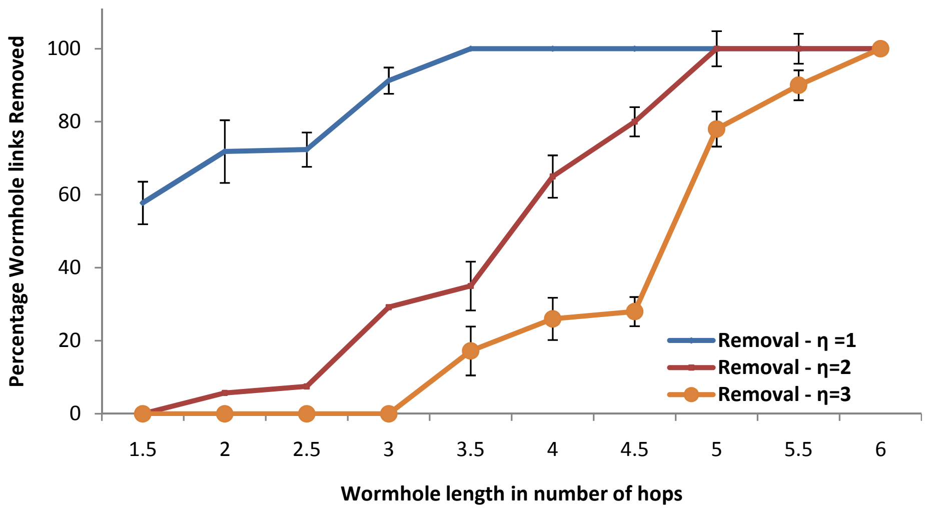

Figure 4.26: Impact of wormhole length on Wormhole Removal

Similarly, the impact of the length of the wormhole on the removal process is shown in Figure 4.26. The results show that the removal process will be enhanced if the wormholes are longer. Note that, the longer the wormhole is, the greater are the number flows attracted to the wormhole and thus greater is the impact on the network. The goal of the attacker is thus to have longer wormholes. In [40] only wormholes of lengths of 6 and 10 hops were considered.

4.4.2.4 Impact of Node Degree To study the impact of node degree on the performance of the secure neighbor creation protocol we have simulated networks with various average node degrees. The value of η is fixed for all the cases in this section to 3. The average node degree was changed by changing the transmission range of the nodes from 110 to 160 meters. Obviously the larger the transmission range the higher is the number of nodes that can be reached by a given node and hence the higher is the average node degree.

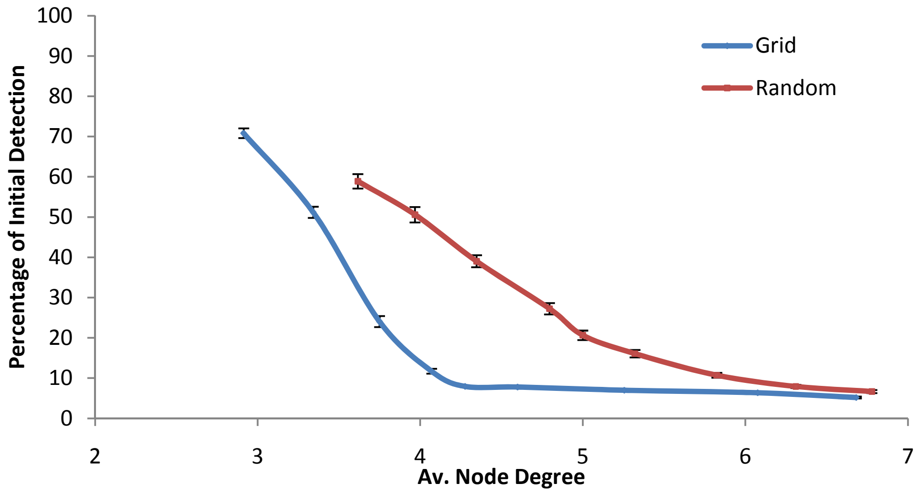

Figure 4.27: Impact of Node Degree on Initial Detection

The impact of average node degree on the false positives caused by the initial detection operation for both grid and randomly distributed networks is shown in Figure 4.27. The results show that the performance of the initial detection process is improved with higher node degree as the percentage of false positives is decreased. This is due to the fact that networks with low average node degrees will have low connectivity which means that it is more likely that the neighbors of a node will have relatively long routes to reach a target node and thus cause false positives. However, from Figure 4.27 it can also be concluded that the initial detection process does not require the network to have a very high node degree to reach a reasonable stable value. An average node degree that is > 4 and > 5 is enough to reach a reasonable stable value of false positives in grid and randomly distributed networks, respectively.

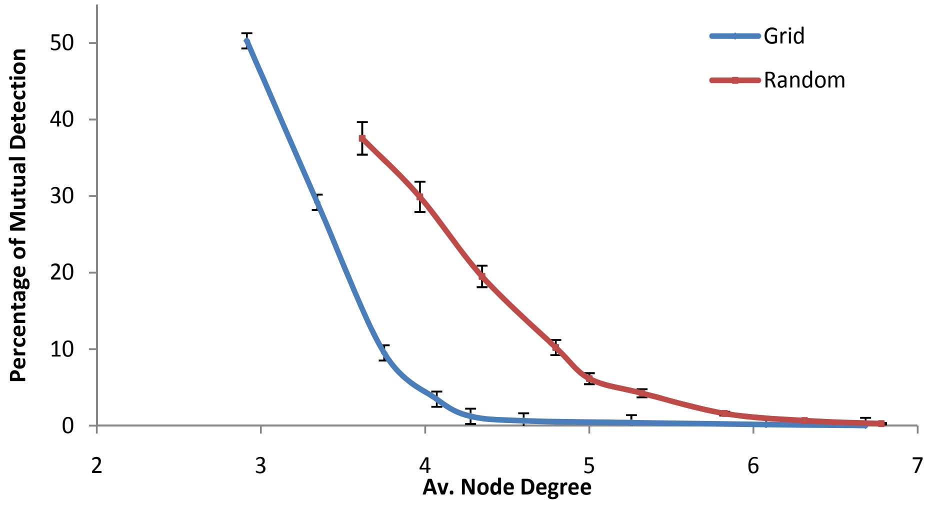

Figure 4.28: Impact of Node Degree on Mutual Detection

Figure 4.28 is similar to Figure 4.27 but with mutual detection, which represents the actual false positive rates caused by the detection process. As expected the percentage of false positives is significantly decreased with the use of mutual detection. A near 0% false positive is reached with an average node degree ≥ 4.5 in grid networks and ≥ 6 in randomly distributed networks. The reason why networks with low node degree have a higher percentage of false positives with mutual detection is that the initial detection results high false positivies for these networks.

The impact of average node degree on the percentage of legal links removed by mistake for both grid and randomly distributed networks is presented in Figure 4.29. The trend in this case is similar to the false positive rates with the detection process shown in Figure 4.28. When two nodes mistakenly mutually detect that they reach each others through a wormhole then it is more likely that the removal process may mistakenly remove some legal

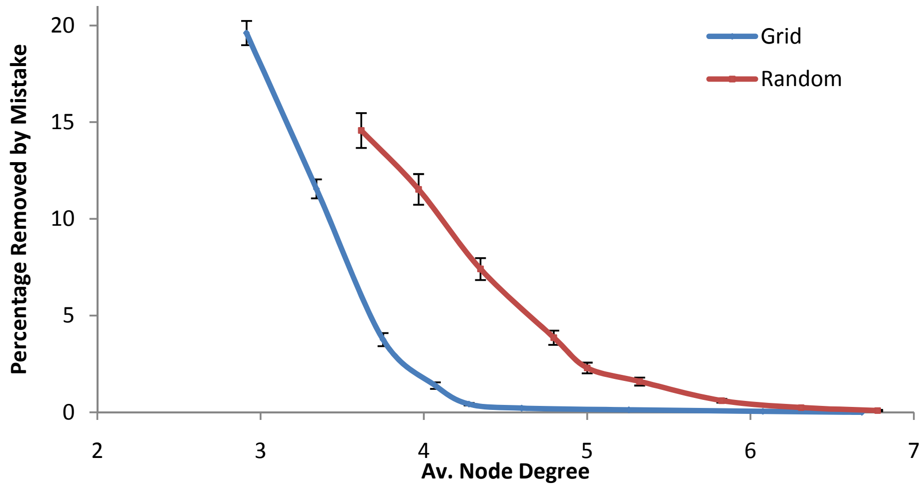

Figure 4.29: Impact of Node Degree on Links Removed by Mistake

links. Obviously, the higher the node degree is, the better is the network connectivity and thus the less likely it is that the neighbors of a node A will find long routes to the target node. Hence, a smaller number of legal links will be removed by mistake.

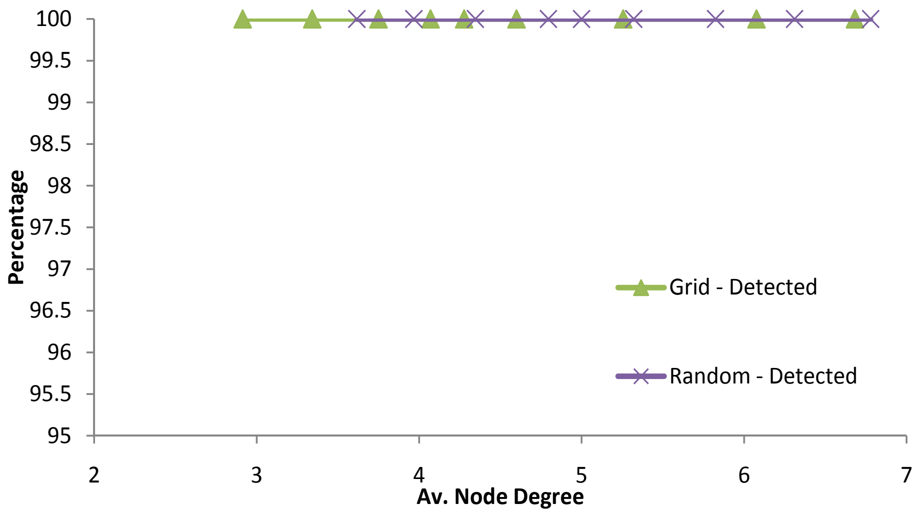

Figure 4.30: Impact of Node Degree on the Detection of Wormhole

The impact of node degree on wormhole detection is shown in Figure 4.30. For both grid and randomly distributed networks, the results show that the detection process can detect wormholes successfully (high rate) even for networks with very low node degrees. An average node degree of 3 and 3.5 is enough to achieve 100% detection with grid and randomly distributed networks, respectively. With an average node degree less than 3, the network connectivity will be an issue (not every node will be able to have a route to every other node).

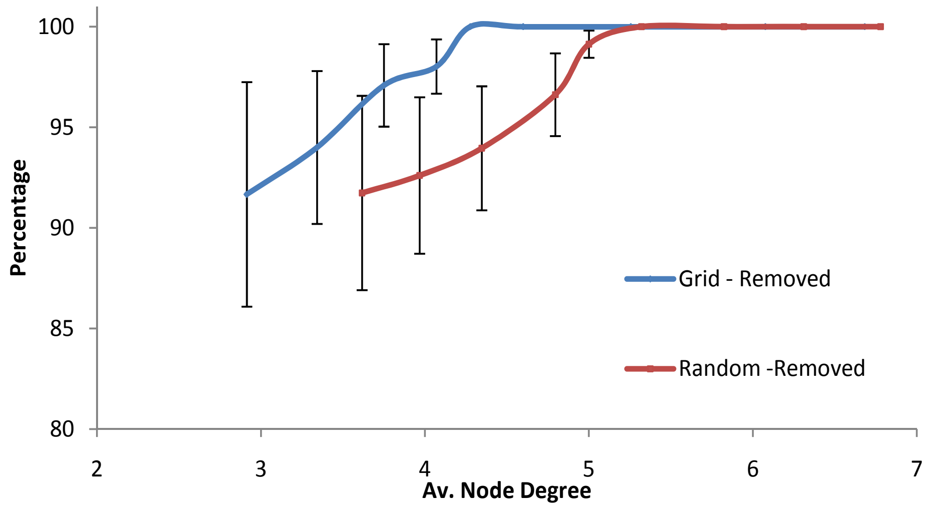

Figure 4.31: Impact of Node Degree on the Removal of Wormhole links

The impact of node degree on the removal of wormhole links is shown in Figure 4.31. The results show that the removal process is improved for networks with higher node degrees. The complete removal of wormhole links is possible with networks of node degree > 4 and > 5 for grid and random distributions, respectively.

4.4.2.5 Networks with Quasi-UDG Connectivity Model We have tested the performance of the secure neighborhood creation protocols for networks with Quasi-UDG connectivity model. As we discussed earlier, the quasi-UDG connectivity is more realistic model, compared to the UDG, as it does not assume that the transmission/reception range of the nodes are unit disks. We have simulated the same network described earlier with 400 nodes, but with quasi-UDG and with two 2-end wormholes.

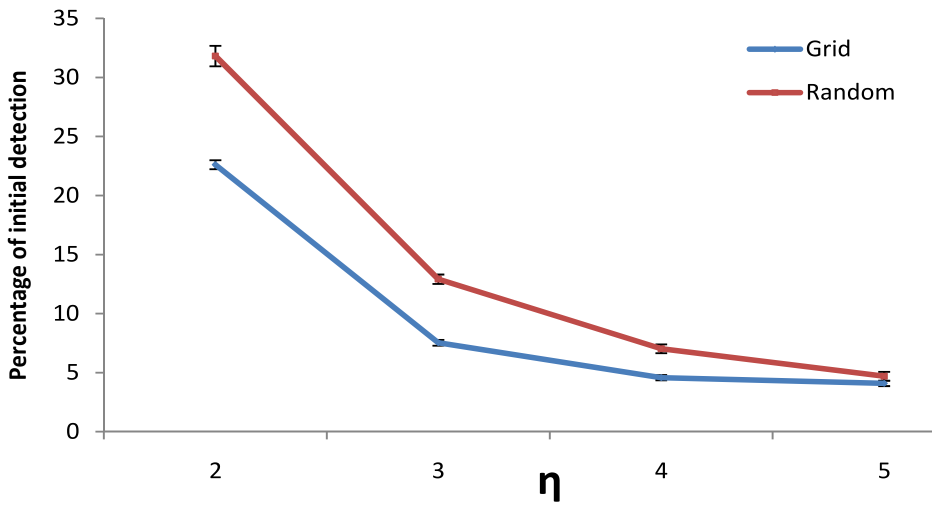

Figure 4.32: False Positives with Initial Detection and Q-UDG

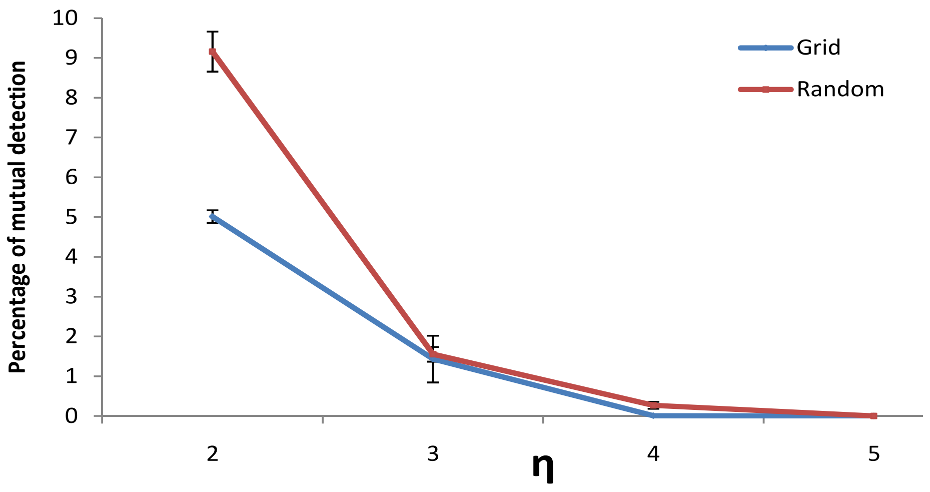

Figure 4.33: False Positives with Mutual Detection and Q-UDG

The percentage of false positives with initial detection operation for networks with quasi-UDG connectivity is shown in Figure 4.32 for both grid and randomly distributed networks. The results show that the percentage of false positives caused by the initial detection process is slightly increased for networks with quasi-UDG connectivity compared to networks with the UDG connectivity model. This is due to the fact that with the quasi-UDG connectivity, not every node at a distance R from node A will be a neighbor, as compared to the UDG. Thus it is more likely that routes from neighbors to a target node will be long enough to cause a false positive during the initial detection operation.

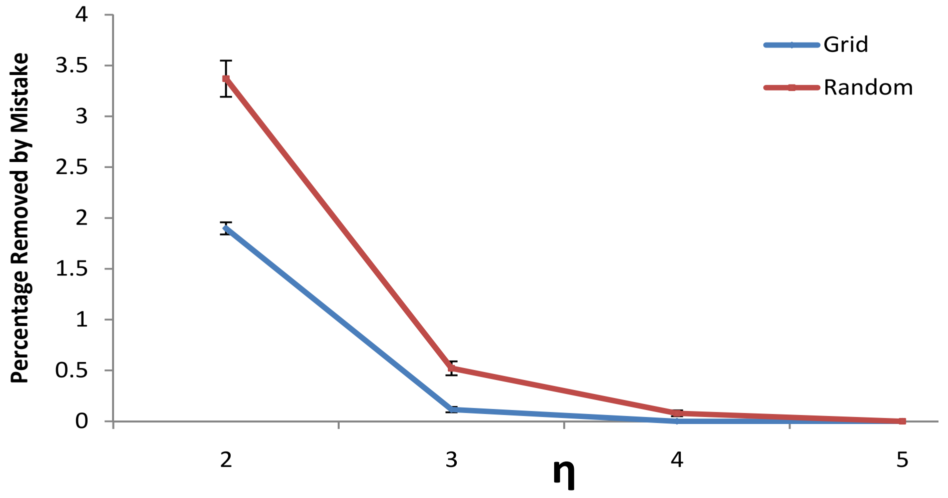

Figure 4.34: Percentage of Legal Links Removed by Mistake with Q-UDG

The percentage of false positives with mutual detection operation for networks with quasi-UDG connectivity is shown in Figure 4.33 for both grid and randomly distributed networks. Again, the results show that the percentage of false positives caused by the mutual detection process is slightly increased compared to networks with a simple UDG connectivity model. This is because the false positives caused by the initial detection operation was also increased.

The percentage of legal links removed by mistake with the removal process for networks with Quasi-UDG is shown in Figure 4.34 for both grid and randomly distributed networks. The results show that the percentage of legal links removed by mistake is still considerably low for networks with quasi-UDG connectivity. It is almost 0.2% and 0.6% with $\eta = 2$ for grid and randomly distributed networks, respectively.

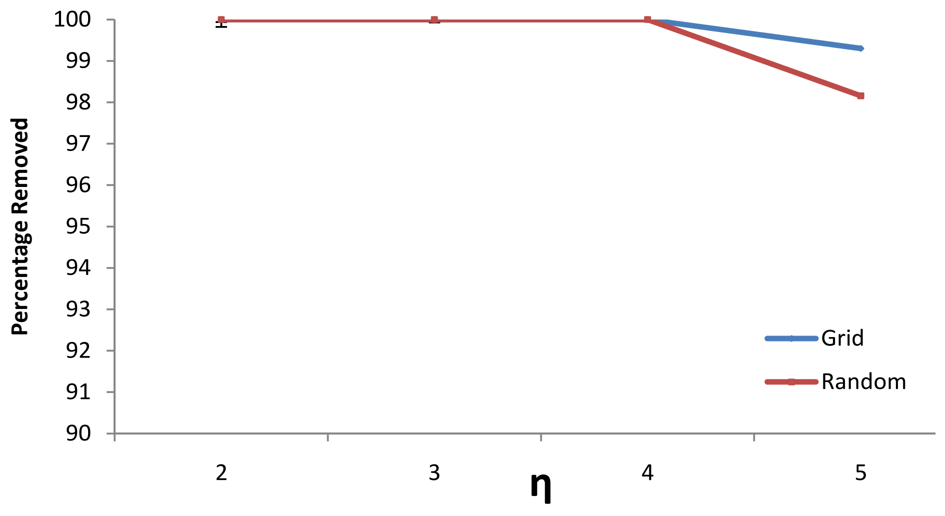

Figure 4.35: Percentage of Wormhole Links Removed with Q-UDG

Figure 4.35 shows the percentage of wormhole links removed by the removal process for networks with quasi-UDG connectivity for both grid and randomly distributed networks. The results show that the percentage of wormhole links removed is 100% for $\eta \leq 4$.

From this section it can be concluded that the secure neighbor creation protocol can still perform very well for networks assuming a quasi-UDG connectivity model.

4.4.2.6 Networks with Asymmetric Links We have tested the performance of the secure neighbor creation protocol for networks with asymmetric links. The asymmetric model that us used here is similar to the one used in the previous chapter. Here we randomly made 50% of the nodes in the network have a shorter transmission range that is 60% of the normal transmission range of the other nodes. We have simulated the same network described earlier with 400 nodes, but with asymmetric links and with two 2-ended wormholes.

The percentage of false positives with initial detection operation for networks with asymmetric links is shown in Figure 4.36 for both grid and randomly distributed networks. The results show that the percentage of false positives caused by the initial detection process is increased for networks with asymmetric links compared to networks with symmetric links.

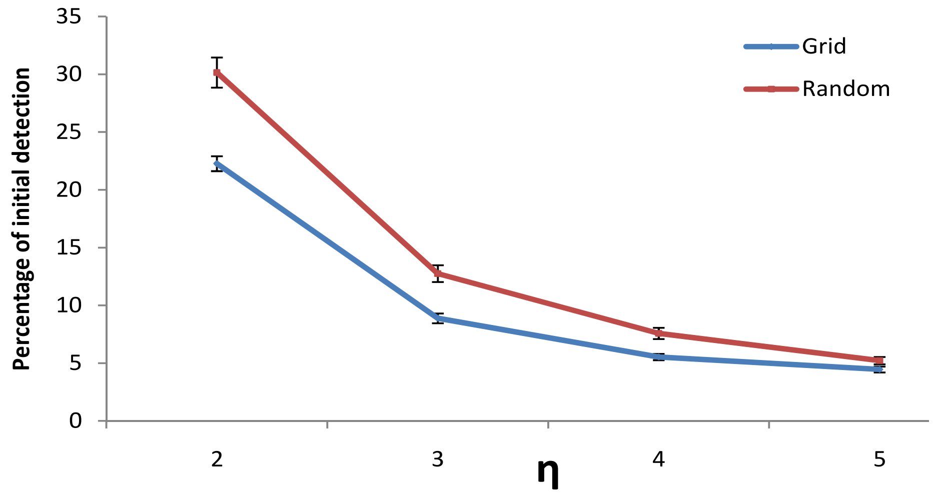

Figure 4.36: False Positives with Asymmetric Links

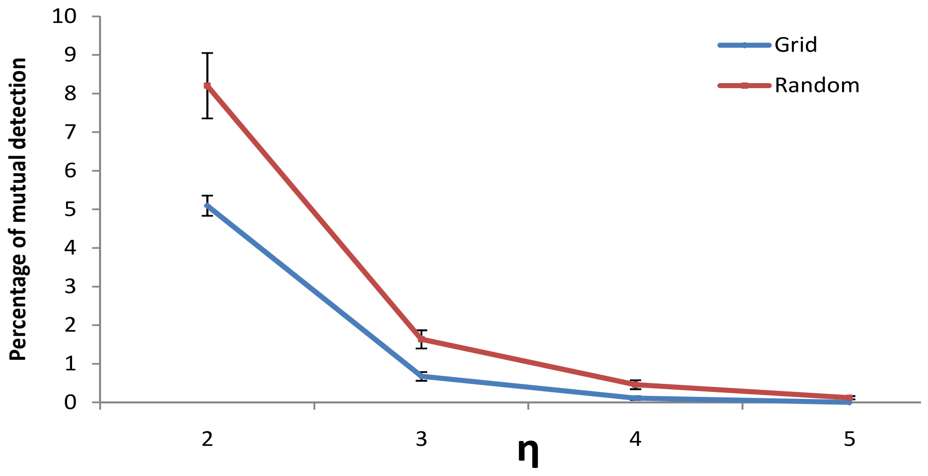

Figure 4.37: False Positives with Asymmetric Links

This is due to the fact that nodes with a short transmission range (60% of normal range) may not be able to reach a neighbor that they could otherwise reach if they had the normal transmission range. Thus it is more likely that routes from neighbors to a target node will be long enough to cause a false positive by the initial detection operation.

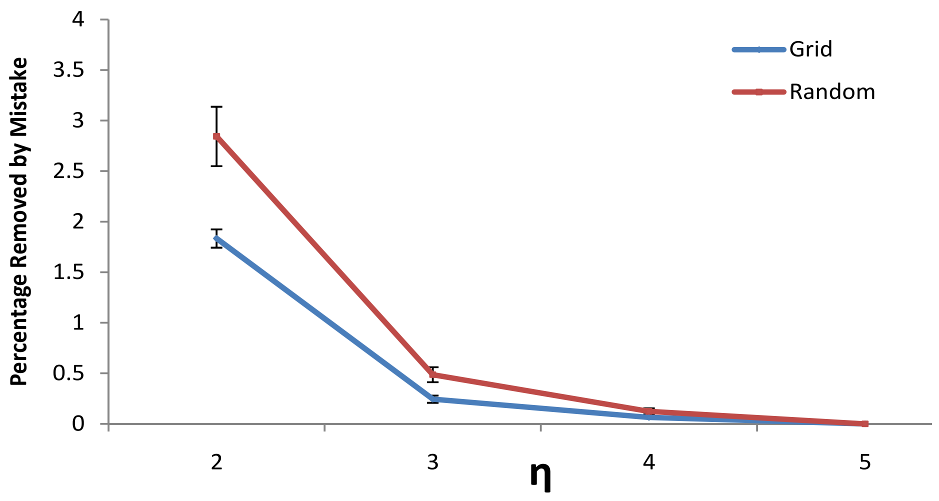

Figure 4.38: Percentage of Legal Links Removed by Mistake with Asymmetric Links

The percentage of false positives with the mutual detection operation for networks with asymmetric links is shown in Figure 4.37 for both grid and randomly distributed networks. Again, the results show that the percentage of false positives caused by the mutual detection process is increased for networks with asymmetric links compared to networks with symmetric links. This is because the initial detection false positive rate was also increased for the asymmetric case.

The percentage of legal links removed by mistake with the removal process for networks with asymmetric links is shown in Figure 4.38 for both grid and randomly distributed networks. The results show that the percentage of legal links removed by mistake is still considerably low for networks with asymmetric links. It is almost 0.3% and 0.6% with $\eta = 2$ for grid and randomly distributed networks, respectively.

Figure 4.39 shows the percentage of wormhole links removed by the removal process for networks with asymmetric links for both grid and randomly distributed networks. The results show that the percentage of wormhole links removed is 100% for $\eta \leq 4$.

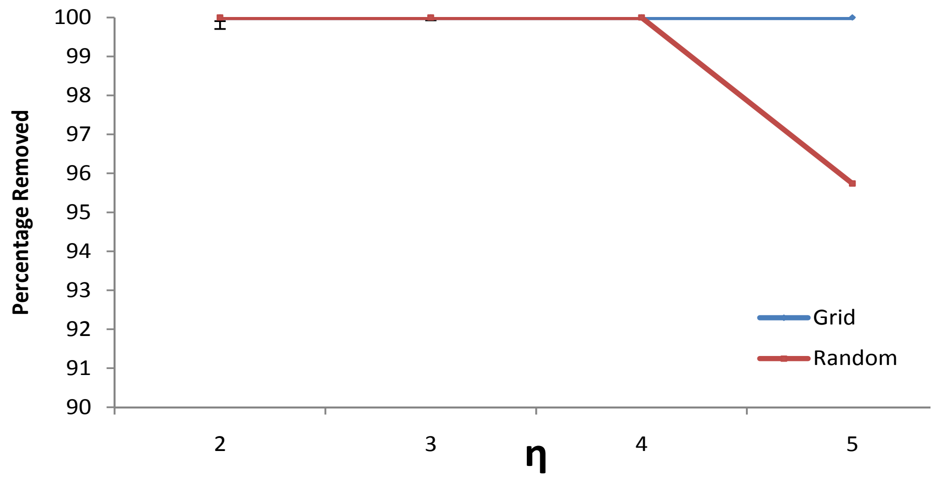

Figure 4.39: Percentage of Wormhole Links Removed with Asymmetric Links

From this section it can be concluded that the secure neighbor creation protocol can still perform very well for networks with asymmetric links as compared to networks with symmetric links.

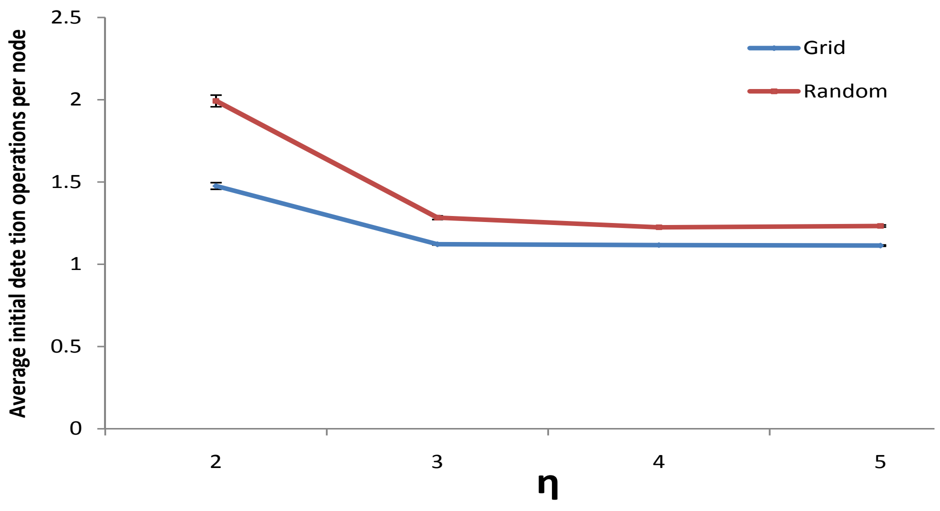

Figure 4.40: Initial Detection Operations Performed per Node

4.4.2.7 Overhead Analysis Here we study the overhead caused by the secure neighbor creation protocol. The main concern here is how many times the initial detection operation is actually performed in the network. This will also impact how many route acquisitions are actually performed in the network. These statistics are accurately collected in the simulations and are presented in this section.

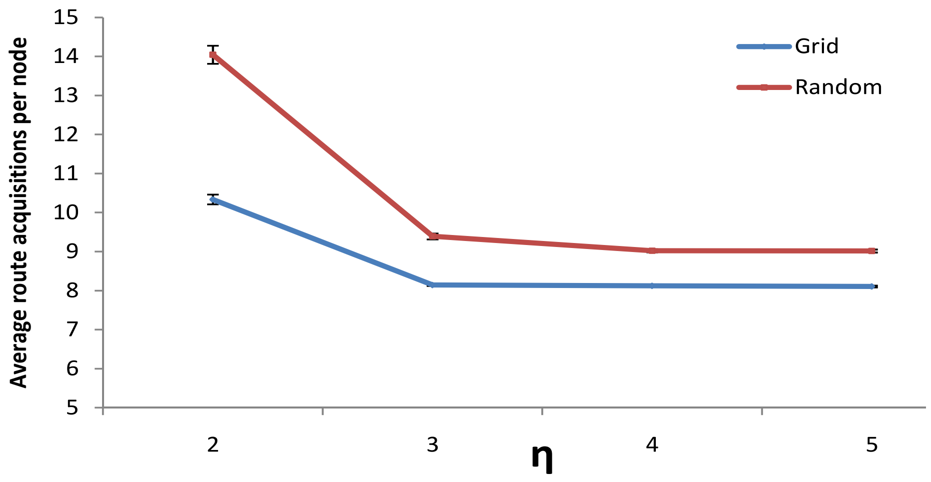

Figure 4.41: Route acquisition per node

Figure 4.40 shows the average number of times each node has actually performed the initial detection operation. The results show that each node will only need perform the initial detection operation very few times. For example, with $\eta = 3$, on average, each node will only need to use the initial detection process with one or at most two neighbors. Note that the average node degree for the network simulated in this case is 6.8. This means that if each node wants to check all its neighbors then each node will on average perform the initial detection 6 or 7 times. However, using the co-operation and control rules we can significantly reduce this overhead. Later in this section we will show the exact overhead savings because of the co-operation and control operation.

The average number of route acquisitions made by each node is shown in Figure 4.41 for both grid and randomly distributed networks. The route acquisition presents the main overhead that is caused by the secure neighbor creation protocol. The results show that the protocol only has a small overhead. For example, with $\eta = 3$ each node will only perform a

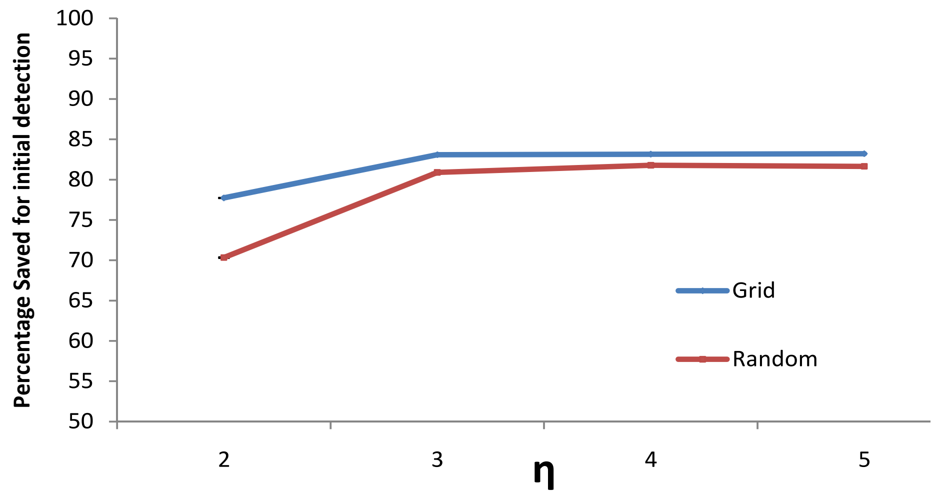

Figure 4.42: Initial Detection Saved using co-operation and Control

total number of route acquisitions that is less than 10. The results in Figure 4.41 show that the nodes in networks with random distribution of nodes require slightly larger numbers of route acquisitions as compared to the nodes in grid networks. This is because the percentage of false positives in randomly distributed networks is higher than in grid networks. Thus, more numbers of initial detection operations need to be performed.

Figure 4.42 shows the percentage of initial detection operations saved by using the co-operation and control operation. This is compared to the case where every node needs to check all its neighbors. The result shows that the co-operation and control operation saves more than 80% of the overhead that could be caused by the initial detection process.

Figure 4.43 shows the percentage of route acquisitions saved by using the co-operation and control rules. This is compared to the case where every node needs to check all its neighbors. The results show that the co-operation and control operation also saves more than 80%, of the overhead in terms of route acquisitions that could be caused by the initial detection process.

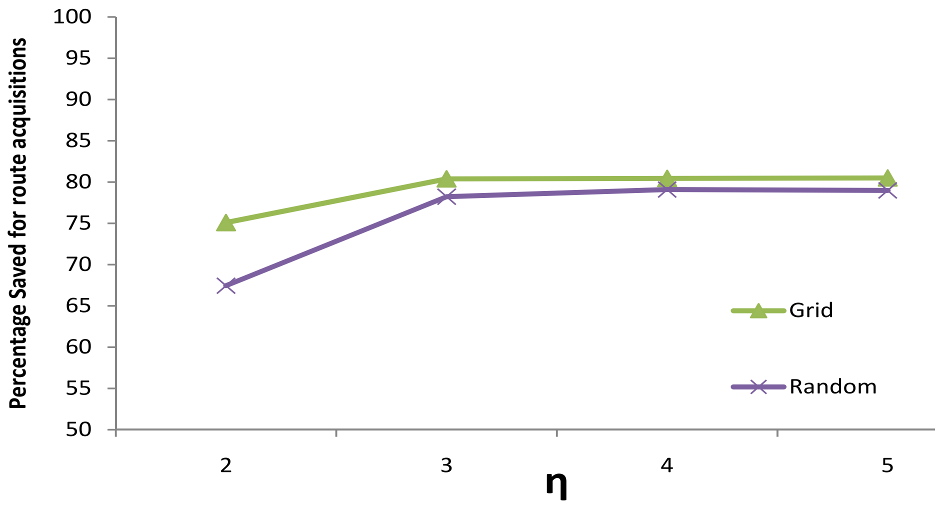

Figure 4.43: Route Acquisitions Saved using co-operation and Control

5.0 CONCLUSIONS AND FUTURE WORK

5.1 CONCLUSIONS

Currently, mobile ad hoc networks are being used in many applications. Most of these applications are critical and require the network to operate securely even in the presence of malicious attackers. The wormhole attack is one of the most serious attacks that threatens the security and stability of ad hoc networks. Additionally, the wormhole attack is considered the main disruptor of neighbor discovery. In fact, many basic ad hoc network functions and operations rely on accurate neighborhood discovery. Consequently, protecting ad hoc networks from wormhole attacks and ensuring secure neighborhood creation is an extremely important issue.

The background material related to security issues and wormhole attacks was presented in Chapter 2 of this dissertation. Definitions, applications, and the most commonly used protocols were all briefly described. Security attacks at various layers in ad hoc networks were also discussed. The wormhole attack was presented in particular details, providing definitions, defining the different types, and explaining the potential impact of such attacks. Neighbor discovery in ad hoc networks was thoroughly discussed. Our own classification for secure neighbor discovery and wormhole detection protocols was presented. The classification was based on the techniques and the approaches the protocols used. The available detection and secure neighbor discovery protocols in the literature were described with clarification of the main issues with each.

Based on the literature review, it was concluded that time-based protocols will not detect physical layer wormholes or wormholes that can relay packets with very small delays. Hence, relying on time as the main tool in the detection protocol is not recommended. Protocols that are based on location information require the nodes to be aware of their location. The nodes have to securely exchange the location information. These protocols require an additional localization or positioning protocol. Hence, they are complicated and the lack of accuracy, especially for indoor applications or urban areas, makes them vulnerable. Protocols that require special hardware are not compatible with all types of nodes, and may require special customization. As we have discussed in Chapter 2, protocols that are based on distance bounding require the use of one or more combinations of time or location information. They may also call for special hardware such as ultrasound. Centralized protocols that need global connectivity information are also not preferred in ad hoc networks without an infrastructure.

In Chapter 3 of this dissertation, we presented DeWorm, a simple protocol for detecting wormholes along routes in ad hoc networks. This protocol employs routing hop count discrepancies between neighbors to determine the existence of a wormhole. Compared to other wormhole detection protocols, the protocol is localized and requires only a small overhead. DeWorm does not need location information, accurate synchronization between nodes, or special hardware. It also does not rely on time measurements. The idea of DeWorm was based on the concept that wormholes aim to create shorter routes in the network that would appear attractive to the nodes. Thus, if the nodes that are connected through the wormhole try to find alternative routes that do not pass through the wormhole, such routes are likely to be much longer thereby revealing the presence of the wormhole. The novelty of the protocol is in how it successfully *avoids* the wormhole in determining the alternative routes by exploiting the end-result of using a wormhole.

In the research literature, researchers have proposed wormhole detection protocols that relied on connectivity or neighborhood information and that did not call for time-measurements, location information, or additional hardware. However, none of these proposed protocols can detect all types and cases of wormholes. Some of them require the wormhole transceivers to connect nodes in a way that must form a forbidden structure in order to be detected. Other protocols that relied on neighborhood information can only detect open wormhole attacks that connect one single visible adversarial node with another single visible adversarial node that is located at least four hops away. This may also work to detect closed wormholes with transceivers that only connect two single honest nodes. However, wormholes usually connect more than one node with another group of nodes. Some other protocols relied on the assumption that a wormhole will significantly increase the node degree of the nodes that it connects. Thus, based on that idea, any node that notices that its node degree is much larger than the average node degree will try to check if a wormhole is connecting any of its neighbors. This may only work if the nodes are uniformly distributed and all the links or edges are fixed. However, in mobile ad hoc networks, the nodes are mobile and randomly distributed. Thus, the nodes will have variable node degrees that could change with time. One of the strongest aspects of DeWorm is that it does not have any of the aforementioned limitations. In fact, as we have discussed in Chapter 3, DeWorm does not have special connectivity or node distribution requirements. The only issues with DeWorm are with *critical nodes* as they could by detected by DeWorm as wormholes. However, simple solutions were proposed to address this problem.

The DeWorm protocol was described in details and with examples in Chapter 3. Intuitively, the protocol was shown to successfully detect wormhole attacks. Different aspects of the protocol were also analyzed such as: the route selected for comparison, the sensitivity parameter η, and the connectivity requirements. The sensitivity parameter plays an important role in DeWorm. Its value will determine the minimum length of wormholes that could be detected by DeWorm. For example, with a sensitivity parameter of 2, any wormhole that is longer than 2 hops will definitely be detected. As we have discussed in this dissertation, wormholes by definition are longer than one hop and attackers always aim to create long wormholes. The longer the wormhole, the greater the impact it will have on the network. Thus, selecting a sensitivity parameter that is less than the length of potential wormholes is the condition for DeWorm to provide 100% detection of wormholes. Other issues such as mobility and using DeWorm with diferent routing protocols were also discussed.

The overhead caused by DeWorm was analyzed mathematically and verified using simulations. An analytical model was developed to be used with given network specifications (such as network size, number of nodes, and nodes' transmission range) to determine the following: (i) the average number of packets that need to be broadcast by the sender nodes (ii) the number of route acquisitions performed between neighbor nodes and target node, which also equals the total number of replies. These values were computed in the simulations and it was shown that they fit the analytical model.

DeWorm was tested through exhaustive simulations for different node distributions, wormhole lengths, and different connectivity models. Under all the evaluated scenarios, DeWorm demonstrated excellent detection rates and few false alarms, revealing its ability to detect wormholes in mobile ad hoc networks. For the node distribution, we have tested DeWorm with grid and randomly distributed networks. Although the results were better with grid networks (lower false positives), DeWorm performed very well under randomly distributed networks. Two connectivity models were used – UDG and Quasi-UDG. DeWorm proved its effectiveness in both. We have also simulated networks with asymmetric links under which DeWorm also performed well. Moreover, we have simulated networks with different node degrees and showed that DeWorm can still achieve strong results for networks with fairly low node degrees, though its effectiveness improves with higher node degrees. Finally, DeWorm was tested with wormholes of different lengths.

In Chapter 4, we presented a secure neighbor creation protocol that can securely discover the neighbors of a node in mobile ad hoc networks by detecting and removing wormhole links if they exist. As with DeWorm, compared to other secure neighbor verification or discovery protocols, our protocol is simple, localized, needs no special hardware, localization, or synchronization. The protocol can also detect and remove multiple two-ended and multi-ended wormholes.

The proposed secure neighbor creation protocol was designed with two main processes: the detection process and the removal process. The first process includes three operations: initial detection, mutual detection, and co-operation and control. The initial detection operation is a modified version of DeWorm. Simulations and analysis showed that it provides 100% detection for all types of wormholes, but with a high number of false positives. The mutual detection operation requires co-operation only between the two nodes that detected the wormhole. The idea is that if a node detected another node to be reached via a wormhole, then the other node must detect the same situation. Simulation results showed that the mutual detection will enhance the performance of the initial detection operation by significantly reducing the number of false positives (to almost 0%) but still providing near perfect detection of wormholes. The co-operation and control operation, manages the use of the previous two operations and requires co-operation between local neighbors (a node and its one-hop neighbors). Instead of having all nodes applying initial detection to all their neighbors (which is the case with other secure neighbor discovery protocols) the co-operation and control operation defined rules that each node must follow and reduced the overall overhead by almost 80%. However, it still maintains a near 100% detection rate and very few numbers of false positives.

The second process of the secure neighbor discovery protocol is the removal process. It consists of two operations: initial removal and mutual removal. This process is the most complicated and depends on the results produced by the detection process. The removal process will only run in locations where wormholes were detected in the detection process. If there are no wormholes detected in the network, the removal process must not remove any link by mistake. However, if there is a wormhole, then it must successfully remove all links between nodes connected using that wormhole. The aim of the mutual removal operation is to enhance the performance of the initial removal operation by reducing the number of links removed by mistake. To our knowledge, this is the first protocol that employs co-operation between the neighbors to reduce the overhead that may be caused if all nodes in the network need to verify all of their neighbors. Also, this is the first protocol that can provide near 100% removal of all the wormhole links and remove only a very few, if any, legal links.

The simulations and analysis were performed for various distributions and connectivity models, similar to the DeWorm simulations. The results showed that the proposed protocol successfully detected and removed all wormhole links. Very few false positives occurred and very few legal links were removed by mistake. The cost was also evaluated and found to be very low, as measured by the number of route acquisitions.

In any research project, there are always limitations and challenges that one wants to further elaborate and examine. As we always have limited time and scope, these issues are usually left for the future work. The following section will present our future work.

5.2 FUTURE WORK

My PhD dissertation focused on detection and removal of wormhole links, which also enables secure creation of neighborhoods. The DeWorm localized wormhole detection protocol uses discrepancies in routing information and it has been evaluated using simulations and analysis. It can be improved if integrated with techniques that use time or location information. This will specifically help to improve detection of extremely short wormholes (1 to 2 hops). Experimental testing and evaluation including such information in real networks can provide more insight into the security implications of wormholes. Analysis of routing discrepancies can also be useful for topology management. Related to topology management is the impact of link removal to disable wormholes (both double and multi-ended wormholes). Statistics about the numbers of links removed and their impact on network connectivity are likely to be useful. Whether a wormhole can be placed to reduce the resilience or robustness of the network through its detection and removal of links is something to be investigated. My secure neighborhood discovery protocol achieves removal of wormholes while minimizing removal of legitimate links (verified by simulations and explained intuitively). As part of my future work, I expect to develop a mathematical proof and bounds on its performance. I also propose to investigate merging/embedding wormhole detection with a secure routing protocol and to implement such schemes for evaluation in a testbed. I expect this work to provide useful results that can enable secure design of ad hoc networks.

Furthermore, mobility was qualitatively discussed and briefly analyzed in this dissertation. It will be interesting to actually test the proposed protocols using simulations with different mobility models. This will also verify our assumption that the mobility (speed) of ad hoc networks will not impact the performance of DeWorm. Thus, testing DeWorm with ad hoc networks with different speeds of mobility will be useful. If possible, I would like to add mobility to the experimental testing of the protocols.

As we discussed in this dissertation, in addition to the wormhole attack, the jamming attack is another threat for neighbor discovery, particularly, limited-range jammers are hard to detect. I would like to expand the proposed model and consider detection of jamming attacks. This will make the proposed secure neighbor creation protocol more complete and comprehensive.

BIBLIOGRAPHY

[1] L. Hu and D. Evans, “Using directional antennas to prevent wormhole attacks,” in *Network and Distributed System Security Symposium (NDSS)*, San Diego, 2004.

[2] N. Sastry, U. Shankar, and D. Wagner, “Secure verification of location claims,” in *Proceedings of the 2nd ACM workshop on Wireless security.* San Diego, CA, USA: ACM, 2003, 941313 1-10.

[3] S. Capkun, L. Buttya’n, and J.-P. Hubaux, “Sector: secure tracking of node encounters in multi-hop wireless networks,” in *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks.* Fairfax, Virginia: ACM, 2003, 986862 21-32.

[4] R. Maheshwari, J. Gao, and S. R. Das, “Detecting wormhole attacks in wireless networks using connectivity information,” in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, J. Gao, Ed., 2007, pp. 107–115.

[5] Y.-T. Hou, C.-M. Chen, and B. Jeng, “Distributed detection of wormholes and critical links in wireless sensor networks,” in *Proc. of IIHMSP*, 2007.

[6] C. Lee and J. Suzuki, “Swat: A decentralized self-healing mechanism for wormhole attacks in wireless sensor networks,” *In Y. Xiao, H. Chen and F. Li (eds.) Handbook on Sensor Networks, Chapter 24, World Scientific Publishing, ISBN: 978-981-283-730-1*, 2010.

[7] T. X. Brown, J. E. James, and A. Sethi, “Jamming and sensing of encrypted wireless ad hoc networks,” in *In Proc. of ACM MobiHoc*, 2006.

[8] H. Yang, H. Luo, F. Ye, S. Lu, and L. Zhang, “Security in mobile ad hoc networks: challenges and solutions,” *IEEE Wireless Communications*, vol. 11, no. 1, pp. 38–47, Feb 2004.

[9] Y. C. Hu, A. Perrig, and D. B. Johnson, “Packet leashes: a defense against wormhole attacks in wireless networks,” in *In Proce. of IEEE INFOCOM*, 2003.

[10] L. Buttyan and J.-P. Hubaux, *Security and Cooperation in Wireless Networks: Thwarting Malicious and Selfish Behavior in the Age of Ubiquitous Computing.* Cambridge University Press, 2007.

[11] J. T. Chiang, J. J. Haas, Y.-C. Hu, P. R. Kumar, and J. Choi, "Fundamental limits on secure clock. synchronization and man-in-the-middle detection in fixed wireless networks," in *Proc. of IEEE INFOCOM*, 2009.

[12] A. Sikora and V. Groza, "Coexistence of ieee802.15.4 with other systems in the 2.4 ghz-ism-band," in *In Proceedings of the IEEE Instrumentation and Measurement Technology Conference*, 2005.

[13] S. Kumar, V. S. Raghavan, and J. Deng, "Medium access control protocols for ad-hoc wireless networks: a survey." *Ad-Hoc Networks*, vol. 4, no. 3, pp. 326–358, May 2006.

[14] M. Abolhasan, T. Wysocki, and E. Dutkiewicz, "A review of routing protocols for mobile ad hoc networks," *Ad Hoc Networks*, vol. 2, no. 1, pp. 1–22, 2004.

[15] B. Wu, J. Chen, J. Wu, and M. Cardei, *Network Theory and Applications.* Springer, 2007, vol. 17, ch. A Survey of Attacks and Countermeasures in Mobile Ad Hoc Networks.

[16] K. Panyim, T. Hayajneh, P. Krishnamurthy, and D. Tipper, "On limited-range strategic/random jamming attacks in wireless ad hoc networks," in *In Proceedings of the IEEE LCN Workshop on Security in Communication Networks*, 2009.

[17] S. Radosavac, A. A. Cárdenas, J. S. Baras, and G. V. Moustakides, "Detecting ieee 802.11 mac layer misbehavior in ad hoc networks: Robust strategies against individual and colluding attackers," *J. Comput. Secur.*, vol. 15, no. 1, pp. 103–128, 2007.

[18] R. Doomun, T. Hayajneh, P. Krishnamurthy, and D. Tipper, "Secloud: Source and destination seclusion using clouds for wireless ad hoc networks," in *In Proceedings of the IEEE Symposium on Computer and Communications*, 2009.

[19] P. Papadimitratos and Z. J. Haas, "Secure routing for mobile ad hoc networks," in *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, 2002.

[20] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer, "A secure routing protocol for ad hoc networks," in *Proceedings of the 10th IEEE International Conference on Network Protocols.* IEEE Computer Society, 2002, 656326 78-89.

[21] M. Khabbazian, H. Mercier, and V. K. Bhargava, "Nis02-1: Wormhole attack in wireless ad hoc networks: Analysis and countermeasure," in *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, H. Mercier, Ed., 2006, pp. 1–6.

[22] R. Poovendran and L. Lazos, "A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks," *Wirel. Netw.*, vol. 13, no. 1, pp. 27–59, 2007, 1228615.

[23] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," in *In First IEEE International Workshop on Sensor Network Protocols and Applications*, 2003, pp. 113–127.

[24] L. Lazos and R. Poovendran, "Serloc: secure range-independent localization for wireless sensor networks," in *WiSe '04: Proceedings of the 3rd ACM workshop on Wireless security*. New York, NY, USA: ACM, 2004, pp. 21–30.

[25] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz, "Localization from mere connectivity," in *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*. New York, NY, USA: ACM, 2003, pp. 201–212.

[26] P. Kruus, D. Sterne, R. Gopaul, M. Heyman, B. Rivera, P. Budulas, B. Luu, T. Johnson, N. Ivanic, and G. Lawler, "In-band wormholes and countermeasures in olsr networks," *Securecomm and Workshops, 2006*, pp. 1–11, 28 2006-Sept. 1 2006.

[27] X. Su and R. V. Boppana, "On mitigating in-band wormhole attacks in mobile ad hoc networks," in *Communications, 2007. ICC '07. IEEE International Conference on*, R. V. Boppana, Ed., 2007, pp. 1136–1141.

[28] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Rushing attacks and defense in wireless ad hoc network routing protocols," in *WiSe '03: Proceedings of the 2nd ACM workshop on Wireless security*. New York, NY, USA: ACM, 2003, pp. 30–40.

[29] W. Wang, B. Bhargava, Y. Lu, and X. Wu, "Defending against wormhole attacks in mobile ad hoc networks: Research articles," *Wirel. Commun. Mob. Comput.*, vol. 6, no. 4, pp. 483–503, 2006, 1144444.

[30] J. Eriksson, S. V. Krishnamurthy, and M. Faloutsos, "Truelink: A practical countermeasure to the wormhole attack in wireless networks," in *Network Protocols, 2006. ICNP '06. Proceedings of the 2006 14th IEEE International Conference on*, S. V. Krishnamurthy, Ed., 2006, pp. 75–84.

[31] I. Khalil, S. Bagchi, and N. B. Shroff, "Liteworp: Detection and isolation of the wormhole attack in static multihop wireless networks," *Comput. Netw.*, vol. 51, no. 13, pp. 3750–3772, 2007, 1276793.

[32] ——, "Mobiworp: Mitigation of the wormhole attack in mobile multihop wireless networks," *Ad Hoc Netw.*, vol. 6, no. 3, pp. 344–362, 2008, 1328997.

[33] A. Boukerche, H. Oliveira, E. Nakamura, and A. Loureiro, "Secure localization algorithms for wireless sensor networks," *IEEE Communications Magazine*, vol. 46, no. 4, pp. 96–101, April 2008.

[34] J.-P. Sheu, C.-M. Chao, and C.-W. Sun, "A clock synchronization algorithm for multihop wireless ad hoc networks," in *In Proceedings of the 24th International Conference on Distributed Computing Systems*, 2004.

[35] K. Sun, P. Ning, and C. Wang, "Secure and resilient clock synchronization in wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 395–408, Feb. 2006.

[36] M. Poturalski, P. Papadimitratos, and J.-P. Hubaux, "Secure neighbor discovery in wireless networks: formal investigation of possibility," in *In Proceedings of the ACM symposium on Information, computer and communications security*, 2008.

[37] P. Papadimitratos, M. Poturalski, P. Schaller, P. Lafourcade, D. Basin, S. Capkun, and J.-P. Hubaux, "Secure neighborhood discovery: a fundamental element for mobile ad hoc networking," in *Proc. of IEEE Communications Magazine*, 2008.

[38] H. Yih-Chun, A. Perrig, and D. B. Johnson, "Wormhole attacks in wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 2, pp. 370–380, 2006.

[39] W. Wang and B. Bhargava, "Visualization of wormholes in sensor networks," in *Proceedings of the 3rd ACM workshop on Wireless security.* Philadelphia, PA, USA: ACM, 2004, 1023657 51-60.

[40] L. Qian, N. Song, and X. Li, "Detection of wormhole attacks in multi-path routed wireless ad hoc networks: a statistical analysis approach," *J. Netw. Comput. Appl.*, vol. 30, no. 1, pp. 308–330, 2007, 1238698.

[41] F. Nait-Abdesselam and T. Tarik, "Detecting and avoiding wormhole attacks in wireless ad hoc networks," *IEEE Communications Magazine*, vol. 46, no. 4, pp. 127–133, April 2006.

[42] A. A. Pirzada and C. McDonald, "Detecting and evading wormholes in mobile ad-hoc wireless networks," *International Journal of Network Security*, vol. 3, no. 2, pp. 191–202, September 2008.

[43] L. Buttyan, L. Dora, and I. Vajda, "Statistical wormhole detection in sensor networks," in *In Proceedings of the Second European Workshop: Security and Privacy in Ad-hoc and Sensor Networks*, 2006.

[44] W. Znaidi, M. Minier, and J.-P. Babau, "Detecting wormhole attacks in wireless networks using local neighborhood information," in *Proc. of IEEE PIMRC*, 2008.

[45] H. Vu, A. Kulkarni, K. Sarac, and N. Mittal, "Wormeros: A new framework for defending against wormhole attacks on wireless ad hoc networks," in *In Proc. of the Third International Conference on Wireless Algorithms, Systems, and Applications*, 2008.

[46] H. S. Chiu and K.-S. Lui, "Delphi: wormhole detection mechanism for ad hoc wireless networks," in *In Proc. of the First International Symposium on Wireless Pervasive Computing*, 2006.

[47] P. V. Tran, L. X. Hung, Y.-K. Lee, S. Lee, and H. Lee, "Ttm: An efficient mechanism to detect wormhole attacks in wireless ad-hoc networks," in *In Proc. of IEEE CCNC*, 2007.

[48] X. Wang and J. Wong, "An end-to-end detection of wormhole attack in wireless ad-hoc networks," in *In Proc. of International Conference on Computer Software and Applications*, 2007.

[49] R. Shokri, M. Poturalski, G. Ravot, P. Papadimitratos, and J.-P. Hubaux, "A practical secure neighbor verification protocol for wireless sensor networks," in *Proc. of ACM WiSec*, 2009.

[50] R. A. Santosa, B.-S. Lee, C. K. Yeo, and T. M. Lim, "Distributed neighbor discovery in ad hoc networks using directional antennas," in *In Proceedings of the Sixth IEEE International Conference on Computer and Information Technology*, 2006.

[51] G. Jakllari, W. Luo, and S. V. Krishnamurthy, "An integrated neighbor discovery and mac protocol for ad hoc networks using directional antennas," in *In Proceedings of the Sixth IEEE International Symposium on World of Wireless Mobile and Multimedia Networks*, 2005.

[52] F. Kuhn and A. Zollinger, "Ad-hoc networks beyond unit disk graphs," in *Proceedings of the 2003 joint workshop on Foundations of mobile computing.* San Diego, CA, USA: ACM, 2003, 941089 69-78.

[53] D. Goyal and J. J. Caffery, "Partitioning avoidance in mobile ad hoc networks using network survivability concepts," in *ISCC '02: Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02).* Washington, DC, USA: IEEE Computer Society, 2002, p. 553.

[54] B. Milic and M. Malek, "Adaptation of the breadth first search algorithm for cut-edge detection in wireless multihop networks," in *MSWiM '07: Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems.* New York, NY, USA: ACM, 2007, pp. 377–386.

[55] N. Gupta and P. R. Kumar, "A performance analysis of the 802.11 wireless lan medium access control," *Communications in Information and Systems*, vol. 3, no. 4, p. 279304, 2004.

[56] K.-I. Kim and S.-H. Kim, "Effectiveness of reliable routing protocols in mobile ad hoc networks," *Wirel. Pers. Commun.*, vol. 38, no. 3, pp. 377–390, 2006, 1147893.

[57] S. Buchegger and J.-Y. L. Boudec, "Performance analysis of the confidant protocol," in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking &amp; computing.* Lausanne, Switzerland: ACM, 2002, 513828 226-236.

[58] 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, 1994.

[59] D. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks, in mobile computing (t. imielinski and h. korth, eds.)," in *Kluwer Acad. Publ.*, 1996.

[60] K. Kaemarungsi and P. Krishnamurthy, "Properties of indoor received signal strength for wlan location fingerprinting," in *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, P. Krishnamurthy, Ed., 2004, pp. 14–23.

[61] I. Broustis, A. Vlavianos, P. Krishnamurthy, and S. Krishnamurthy, "Ctu: Capturing throughput dependencies in uwb networks," *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pp. 412–420, April 2008.

[62] V. Shah, E. Gelal, and S. V. Krishnamurthy, "Handling asymmetry in power heterogeneous ad hoc networks," *Comput. Netw.*, vol. 51, no. 10, pp. 2594–2615, 2007, 1243143.

[63] T. Hayajneh, P. Krishnamurthy, and D. Tipper, "Deworm: A simple protocol to detect wormhole attacks in wireless ad hoc networks," in *In Proceedings of the IEEE Symposium on Network and System Security*, 2009.