September 2013

Policy-Based Immunization Framework for MANET

Arash Tajalli-Yazdi
The University of Western Ontario

Supervisor
Professor Hanan Lutfiyya
The University of Western Ontario

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science

Policy-Based Immunization Framework for MANET

(Thesis format: Monograph)

Arash Tajalli-Yazdi

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

The School of Graduate and Postdoctoral Studies
The Western University
London, Ontario, Canada

Abstract

Mobility is one of the most important driving forces of the hyper-interconnected world that we are living in. Mobile computing devices are becoming smaller, more ubiquitous and simultaneously providing more computing power. Various mobile devices in different sizes with high computing power cause the emergence of new type of networks' applications. Researchers in conferences, soldiers in battlefields, medics in rescue missions, and drivers in busy highways can perform more efficiently if they can be connected to each other and aware of the environment they are interacting with. In all mentioned scenarios, the major barrier to have an interconnected collaborative environment is the lack of infrastructure. Mobile Ad hoc Networks (MANETs) are very promising to be able to handle this challenge. In recent years, extensive research has been done on MANETs in order to deliver secure and reliable network services in an infrastructure-less environment. MANETs usually deal with dynamic network topologies and utilize wireless technologies, they are very susceptible to different security attacks targeting different network layers. Combining policy-based management concepts and trust evaluation techniques in more granular level than current trust management frameworks can lead to interesting results toward more secure and reliable MANETs.

Keywords. Mobile Ad Hoc Networks, Policy-Based System Management, Trust Managent

Contents

List of Figures

Chapter 1

Introduction

In 2011, for the first time, the global unit shipment of smartphones and tablets surpassed desktop and notebook PCs [36]. Today, mobile smart devices can be seen everywhere and terms like Internet of Things and service oriented architecture are commonly used. Users desire access to their information wherever they are and whenever they want. To move toward this goal, advanced wireless communication standards have been developed, and different types of connected devices have been introduced.

For many applications, these devices need to be connected to an existing network to access data and perform their assigned tasks. Today, a fixed and robust network infrastructure is in place and millions computers are using different types of protocols to make connections between themselves in order to provide a framework for developing and deploying new distributed computing systems. New smart mobile devices have been introduced, and new operating systems and protocols have been developed. These devices can form wireless networks, which are connected to the pre-existing infrastructures. This type of wireless network is called infrastructure-based wireless network and can be seen as an extension to the fixed wired network. In infrastructure-based wireless network [20], wireless devices are all connected to an access point, which acts as a bridge between wired and wireless networks. Infrastructure-based wireless networks form wireless LANs, which are used in offices, airports, and train stations. Deploying this type of network can cut the cost of wiring and maintenance of the network [20]. Figure 1.1 demonstrates the infrastructure-based wireless network.

In some scenarios, smart mobile devices do not have access to any fixed network infrastructures. In order to accomplish an assigned task the devices should be connected to each other in an ad hoc manner. In these scenarios, there is no pre-existing network infrastructure

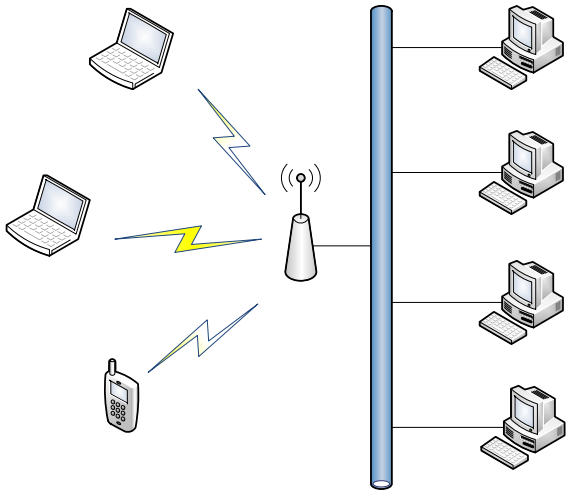

Figure 1.1: Infrastructure based wireless network

to rely on. Responsibility for providing network connectivity is shared by all the mobile devices. For example, in military battle fields, there is no pre-existing network infrastructure; If troops or military equipment need to be connected to each other to conduct a mission, they need to form an ad hoc network. These types of ad hoc networks are infrastructure-less wireless networks. An infrastructure-less wireless network is illustrated in Figure 1.2.

Ad hoc networks are decentralized types of wireless networks. In this type of network, nodes should cooperatively form the network and manage all the connectivity issues. In most cases, it is desirable that the network can operate and accomplish its mission without any central entity. Trying to have a network managing mechanism without any central entity is a step toward having a truly distributed system.

In the majority of ad hoc networks' applications, nodes are not stationary. Nodes change their positions in the network. While they are moving, they need to stay connected to the network. This type of mobile ad hoc networks are categorized as MANET [20]. Lack of centralized management entity, mobility, and wireless communication makes MANET susceptible to a variety of security attacks. With advances in wireless communication and proliferation of mobile devices, researchers try to investigate the different security solutions for MANETs in order to make them capable of delivering reliable services.

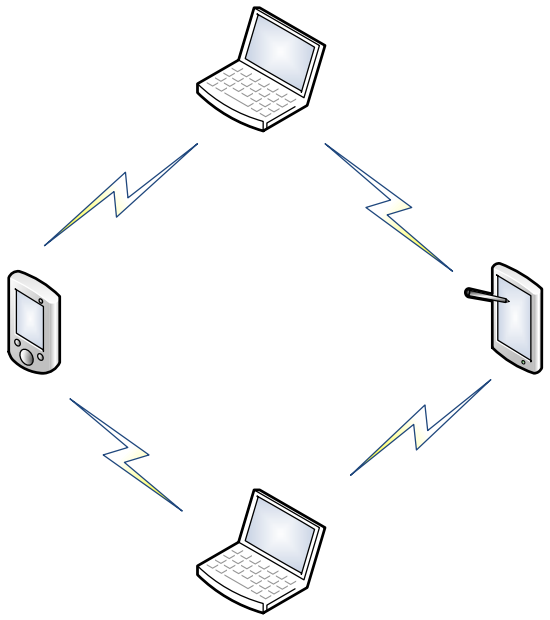

Figure 1.2: Infrastructure less wireless network

1.1 MANET characteristics and applications

Mobile ad hoc networks are a collection of mobile devices, equipped with wireless technologies that form a network without any central administrative entity [20]. In this type of network, nodes can communicate with each other directly or through other nodes without using any central entity to manage the connections. In a MANET, nodes acts as hosts and routers. Node mobility causes frequent network topology changes and routing protocols should be capable of maintaining connectivity in such a dynamic environment. Some of the most important characteristics of MANET are as follows [20]:

- As a result of wireless communication links, MANETs have lower bandwidth capacity in comparison to wired networks; they are subject to jamming, signal fading and interference [49];
- In MANETs, nodes usually depend on battery power. Thus nodes should perform their tasks in the most energy efficient manner;
- Because of wireless links and absence of central entities, security in MANETs can easily be compromised. Wireless nodes are vulnerable to eavesdropping and jamming. Therefore, detecting malicious from benevolent nodes is not easy;

- Mobility of nodes lead to frequent topology changes. Routing protocols in MANETs should be agile in responding in order to frequent changes to maintain a current route between nodes [2];

- Because of lacking a fixed-infrastructure, MANETs are easy and fast to deploy. Therefore, they can provide an instant network formation, and reduce the cost of deployment and maintenance of the network;

These characteristics make MANET an ideal option for the following applications

- MANET can be used in military battlefields, where there is no fixed infrastructure and nodes (troops or military equipments) need to be connected to each other;

- MANET can be used in rescue operations, where network infrastructures are damaged due to natural catastrophes;

- MANET can be used in classrooms, conferences, and meetings in order to allow participants to share their information without the need of connecting to a central server;

- MANET can be used to form personal area networks [47], where a person wants to form a network between his/her PDA, laptop, and a printer without having to plug anything in;

1.2 Security challenges in MANET

Securing MANETs, because of salient characteristics of mobile ad hoc networks such as vulnerability of channel, vulnerability of nodes, absence of infrastructure, and dynamically changing topology is a challenging task [19]. Moreover, every application has its own security requirements. A robust and reliable security solution should provide security services such as authentication, confidentiality, integrity, anonymity, and availability. Here we briefly describe these services

Availability means that a node should provide all promised services regardless of security state of it [30]. Sometimes nodes in a MANET become malicious or selfish and hesitate to collaborate with other nodes to deliver the designed service of the network. This is the main root of denial of service [49] attacks.

Integrity of messages should be kept during transmission. Either malicious activities or transmission disturbance can corrupt a message. Integrity guarantees that a transmitted message is not corrupted [19].

Confidentiality means that information can be accessed by those who have been authorized to access it [30]. It guarantees that sensitive information like military, routing or personal information is never disclosed to unauthorized entities [19].

Authenticity ensures that claims of identity from participants in a communication are authentic [30]. If there is no such a service in place, a malicious node can impersonate a benign node and start a communication with another node.

Non-repudiation ensures that a sender or receiver of a message can not deny that they have ever sent or received such a message [30].

Authorization is the process of issuing credentials which specify the privileges and permissions of accessing a resource. Authorization is used to assign different access rights to users from different user levels [49].

Anonymity means that a node should privately keep all the information that can be used to identify the owner or the current user of the node and should not distribute this information in the network [30].

Several attacks have been designed to violate these security services [49]. Attacks target different network protocols and attempt to disrupt the proper functionality of the network. In order to design a security solution for MANETs, as noted by V. Balakrishnan et al. [6], maybe the fundamental question to be addressed is "*how to enable a mobile node to enlist trusted intermediate mobile nodes so that they can cooperate in forwarding the information to a target without modifying the information or obstructing the operation of other mobile nodes.*" This shows that security of MANETs heavily depends on security of communication layer.

In order to deliver security services in MANETs, several secure routing protocols like SAODV [32] have been suggested. However because of the absence of centralized authority, key management which is the key function of secure routing could not be done properly. Also, because of repeated topology changes secure routing protocols are incapable of distinguishing

genuinely broken links from maliciously reported ones [6]. Hence, incentive-based systems have been proposed. Incentive-based systems such as SIP try to stimulate cooperation in packet forwarding among nodes by introducing concepts like credit [50]. Incentive-based systems to some extent could motivate selfish nodes, which do not collaborate with other nodes in order to save their battery power, CPU cycle, or available bandwidth, to cooperate in forwarding the network traffic with other nodes. In contrast, malicious nodes which deliberately want to disturb the functionality of the network can still ignore all the incentives and harm the network. Shortcomings of secure routing and incentive-based protocols led to emergence of trust-based management systems [6] . Trust-based management solutions introduced the concept of trust among nodes in MANETs. These systems usually consist of three main modules: *monitor*, *reputation manager*, and *trust manager*. By utilizing these modules, they can proactively detect and reactively isolate malicious or selfish nodes [6]. Several trust-management systems have been introduced [13][18] but few of these use their acquired information from network monitoring and knowledge of an attack's behaviour toward defining policies for detection of a specific type of attack. Doing that, more effective policies will be defined and attacks will be detected with higher rates.

1.3 Thesis focus

The thesis focus is to introduce a new trust management system that can be deployed on nodes in mobile ad hoc networks and make them resistant to security attacks. The computation power and network bandwidth of nodes are limited in MANET, thus we plan to develop a computationally light-weighted mechanism to turn each node into a autonomous abnormality detector. We will show how our framework utilizes policies and a simple collaboration scheme among neighbours to prevent potential abnormalities.

1.4 Thesis organization

The remainder of this thesis is organized as follows: Chapter 2 covers background and related work, including key definitions, concepts and a review of the current research relevant to security issues in Mobile ad hoc networks, policy-based management systems, and trust management in MANET. Chapter 3 describes the architecture of our proposed policy based trust management system to be deployed in each node in Manet. Chapter 4 describes the implemented prototype of our proposed framework. Chapter 5, Experiment, defines a case

study and shows how the framework can handle abnormality detection in Manet. Finally, Chapter 6, Conclusion, provides some final conclusions and presents ideas and thoughts for future research in the area.

Chapter 2

Background and Related Works

This chapter presents key definitions and concepts, and reviews the current research which is relevant to MANETs' security vulnerabilities and attacks. Section 2.1 presents security issues in MANET and discusses possible security attacks in different network layers. Section 2.2 presents different types of routing which are being used as the bases of mobile routing algorithms. Section 2.3 describes common routing protocols in MANET. Section 2.4 explains the notion of policy in managing a MANET. Finally, in section 2.5, a wormhole attack and its countermeasures are studied.

2.1 Security in Mobile Ad hoc Networks

Attacks in mobile ad hoc networks can be classified from different perspectives. Each attack can be classified as *Internal* or *External* and *Passive* or *Active* [19].

2.1.1 Internal vs. External attacks

External attacks are carried out by outsiders. These types of attacks are launched by nodes that do not belong to the domain of the network. On the other hand, *internal* attacks are performed by compromised insiders that are network members and have some privileged access rights that make them more disruptive and hard-to-detect.

2.1.2 Passive vs. Active attacks

Passive attacks obtain information from communications in the networks without disrupting the normal behaviour of the network. Eavesdropping, traffic analysis, and traffic monitoring

[49] are examples of passive attacks. On the other hand, *active* attacks not only obtain information from the network, they also modify, fabricate, drop, or replay packets in the network. Thus, they interrupt the normal behaviour of the networks. *Jamming*, *spoofing*, *modification*, *replaying* are examples of active attacks [49].

Wu, Bing, et. all in [49] further categorized MANET attacks based on different layers, *stealthy* versus *non-stealthy* attacks, and *cryptography* versus *non-cryptography* related attacks.

2.1.3 Security attacks in mobile ad hoc networks

2.1.3.1 Physical layer attacks

The broadcast wireless nature of MANETs' communications means that *signal jamming* and *eavesdropping* can be easily done by an adversary.

- Eavesdropping: Wireless links are broadcast in nature. Some mobile hosts share a wireless medium in ad hoc networks. Thus, wireless signals, which are usually in radio frequency spectrum, can be intercepted and heard by unauthorized third parties. Eavesdropping is the capturing of a message by an unauthorized receiver who is looking for confidential information such as location, public key, private key, or passwords [30].

- Interference and jamming: Radio signals can be jammed and interfered with some stronger signals emitted from outsider or compromised insider nodes. These strong signals can corrupt the messages and prevent messages from being delivered to their destination. The attacker can transmit random noises and pulses into the network which can in turn overwhelm the network signals.

2.1.3.2 Link layer attacks

Link layer protocols are responsible for ensuring the connectivity between 1-hop neighbours. Wireless medium access control (MAC) protocol has to coordinate the transmission of nodes on transmission medium. Since token-passing bus, which is used in wired network can not be used in wireless environment, IEEE 802.11 protocols are dedicated to dealing with controlling the wireless medium [49]. The 802.11 MAC working group suggested two contention resolution mechanisms: *distributed coordination function* and *point coordination function* [7]. Only the decentralized scheme in applicable to MANETs [11]. DCF is a random access

scheme which is based on carrier sense multiple access with collision avoidance (CSMA/CA) protocol. DCF uses handshake scheme called RTS-CTS along with binary exponential back off rules to avoid possible collisions in wireless environments [7]. An attacker can deviate from CSMA/CA rules in a way it can disrupt the network operations toward its own benefits.

2.1.3.3 Network layer attacks

Network layer protocols are responsible for extending the one-hop neighbouring connectivity to all other nodes in the network. There are a large number of attacks targeting this layer in the network. An attacker tries to inject itself to the network in order to disrupt the routing process in different phases of routing such as route discovery, route maintenance, or data forwarding [25]. Network layer attacks take the control of the network traffic flow and cause severe damage such as preventing a node from finding the optimum path to the destination, partitioning the network, forwarding packets to a nonexistent path, and creating routing loops. Sometimes two or more colluding nodes collaborate to perform sophisticated attacks in order to take control of the network traffic flow. Some of the network layer attacks are as follows [49].

- Routing table overflow attack: In this attack, attacker advertises routes to non-existent nodes and causes other nodes to become incapable of finding new routes while having their routing tables overwhelmed by false routes. This attack targets proactive routing protocols because proactive routing protocols rely on periodic route advertisements. In order to attack on-demand routing protocols two or more colluding nodes are needed in a way that one node request a route to nonexistence destination and the other node replies with a false route [16].

- Routing cache poisoning attack: In ad hoc networks, sometimes when a node senses a packet, it updates its routing table using information contained in that packet. This operation is called promiscuous routing table updating. A malicious node can broadcast a packet with false routing information and cause some nodes to update their routing table information with this incorrect information. Doing this, the malicious node can direct the network traffic to pass through it [49].

- Replay attack: Due to node mobility, network topology frequently changes. In this attack, a node records routing control messages of another node and replay them later. Doing this, the node deceives other nodes about routing information that may be stale [25].

- Wormhole attack: A wormhole attack is a severe network layer attack which can disrupt a high percentage of communication across the network. It is easy to implement and challenging to detect. In this attack, a malicious node records packets in one location of a network and tunnels them to another malicious node at a distant point in the network. The second malicious node replays the received packets. Wormhole attacks are discussed in detail in 2.5.

2.1.3.4 Transport layer attacks

The transport layer in MANET is responsible for reliable delivery of packets, congestion control, flow control, and clearing of connections [49] between communicating entities. TCP-like protocols in MANET should handle higher channel error rate in comparison to TCP in weired networks. Like the TCP protocol in fixed networks, MANET protocols are vulnerable to *SYN flooding* and *session hijacking* attacks.

- Session hijacking: In this attack, attacker spoofs the victim's IP address and finds the expected sequence number in a TCP session between the victim and a third entity with an active session with the victim. Using this information, an attacker can hijack the established session and gain the control over the session. Usually by launching a DoS against the victim, the attacker keeps victim busy and uses the hijacked session to communicate with the third entity. The fact that most communications are protected only at session setup not thereafter is used by attacker to launch this attack.

- SYN flooding: In order to establish a TCP connection, two communicating nodes have to do a three-way hand shake [28]. During the hand shake process three messages of type *SYN* and *SYN-ACK* are exchanged. Upon completion of the hand shake process, a fully-opened TCP connection is created.

  An attacker sends a large number of SYN packets with spoofed return addresses to a victim. The victim replies to SYN packets with *ACK* packets and waits for the response of *ACK* from the attacker to complete the three-way hand shake. The attacker does not send a final *ACK* to the victim which leads to many half-open connections and waste of resources of the victim [24].

2.1.3.5 Application layer attacks

The application layer is where all user data, which are an attacker's highest interest, reside. The application layer is the home of protocols such as HTTP, FTP, SMPT which potentially

can provide many vulnerabilities that attackers could have a stake in.

- Malicious code attacks: Malicious code can appear in the form of *viruses*, *worms*, and *Trojan horses* and disrupt the proper functionality of the network. *Viruses* are self-replicating code segments that are attached to other host executables. As soon as the host is executed the virus code is activated. *Viruses* usually replicate themselves and attach to different executables. *Worms* are also self-replicating programs but they do not need any other executable to play role of a host for them. They can run and replicate themselves without any other intervention. *Worms* usually exploit network services to propagate themselves through the network [38]. *Trojan horses* are malicious codes masquerading as a legitimate application. *Trojan horses* do not replicate and they try to deceive their victims that they are providing benign services and then start to capture information.

- Repudiation attacks: Repudiation is the denial of participation in a communication. A secure system has to provide non-repudiation as one of its security services. It means the system should ensure that entities could not deny what they have done in the different states of the system. Repudiation attacks take advantage of lack of a proper logging system and launch repudiation attacks.

2.2 Routing

Routing is the process of selecting a path through which network traffic can be forwarded from one host to another. Routing is the responsibility of the network layer in the network protocol stack. When a node joins a network, it is completely unaware of the network topology. Therefore, it starts a process, called neighborhood discovery, in which the node constructs its routing table and starts to learn about the network topology. After the discovery phase a node starts another process called routing table update in order to maintain its connectivity to the network. In this phase, nodes have their knowledge of topology updated by exchanging their routing tables. There are two broad classes of routing algorithms: decentralized and global [28]. In a decentralized routing algorithm, routing nodes do not have the knowledge of all the links in the network. Each node, through collaboration with its neighbors will eventually have a routing table which is complete and up-to-date. On the other hand, in global routing algorithms, each node has the complete overview of the network's links and it broadcasts all its routing table changes not only to its neighbors, but also to all other nodes.

2.2.1 Distance Vector routing protocols

Distance-vector protocols are decentralized. In these protocols, nodes are not aware of the whole path to destination, and nodes only maintain a vector (table) of destinations and minimum distance to destination nodes in their routing tables. In this approach, nodes periodically, upon expiration of a counter or updates in routing table, send their routing tables to their neighbors. By receiving routing tables from its neighbours, a node can recalculate its routing table and forward the new routing table to its neighbors and so on. Since nodes usually deal only with the routing tables received from their neighbors, the topology control traffic does not use a large chunk of network bandwidth. Also, nodes do not need to consume high processing power and memory in order to have an updated routing table. These type of routing protocols use Bellman-Ford [45] algorithm to calculate paths between communicating nodes, and they are not loop-free. These routing algorithms are susceptible to count-to-infinity problem [31] and suffer from scalability issues. Distance vector routing protocols, in comparison to link-state routing protocols, have less computational complexity and message overhead. RIPv1, RIPv2, and IGRP are examples of distance vector routing protocols [28].

2.2.2 Link-State routing protocols

Link-state routing protocols are global. In link-state routing protocols, routing algorithms need to have a complete knowledge of the network topology. These types of routing algorithms take a complete and global knowledge of connectivity between all nodes in order to compute the least-cost path between source and destination. To do so, nodes need to broadcast the identities and costs of their links to all other nodes. Besides routing table, each node has a topological database to store all the links' information in the network. This information is broadcasted using a specific message called link-state advertisement [3]. This data propagation through the network is called flooding. Initially, a node does not know about all other nodes in the network, but after a while, by receiving other nodes' broadcasts, it eventually learns the topology of the network, and starts the path calculation process. In link-state algorithms, usually Dijkstra's algorithm is used in path-calculation process and count-to-infinity problem does not arise. In comparison with distance-vector algorithms, link-state routing algorithms need more computational power and storage. OSPF, IS-IS are examples of link-state routing protocols [28].

2.3 Routing in mobile ad hoc networks

In ad hoc networks, nodes perform as hosts and routers. Initially, a node does not have any knowledge about other nodes in the network. Nodes need to disseminate information about their existence, and listen to the network in order to find the current topology. Based on the nature of the ad hoc networks, ad hoc routing protocols should deal with an ad hoc network's limitations, such as limited bandwidth, high power consumption, and high error rate. In order to better understand routing protocols for MANETs, the reason we cannot use conventional routing protocols in mobile ad hoc networks should be investigated. The following is a summarized list of MANET characteristics which shows why designing a routing protocol for MANET is a challenging task.

- In MANET rate of topology changes is very high, and it causes difficulty in convergence of conventional routing algorithms into steady state [41];
- Wireless links are less reliable that wired ones, and their performance will fluctuate based on environmental conditions;
- Wireless links can be unidirectional which prevents the proper handshake between sender and receiver;
- Wireless links, lack of centralized entity, dynamic topology changes, and delay in propagation of new routing information makes the design of stable routing protocol for MANETs challenging;
- Wireless devices usually rely on batteries as their source of power. Sometimes a lack of power can force a node to act selfishly and prevent its participation with other nodes to form updated routes.

Thus, routing protocols in ad hoc networks should be capable of dealing with all these characteristics of MANETs and provide a service that can ensure scalability and security. This is why classic routing protocols are inadequate for MANETs. An ideal routing protocol for wireless ad hoc network should have the following characteristics:

- cope with frequent topology changes
- deal with possible unidirectional links
- be simple

- scale well in different network parameters such as network size, mobility, and traffic
- perform in a distributed manner
- handle multiple routes for a specific destination in order to reduce the topology control messages
- be loop free
- prevent network partitioning

2.3.1 MANET routing protocols from different perspectives

MANET routing protocols can be classified based on the way they envision the network, which conventional routing approach they use, and how they act upon topology changes.

2.3.1.1 Flat, Hierarchical, and Geographical routing protocols

Flat routing protocols consider every node to be equal. In this type of routing, when a node wants to find a route to another node, it considers all other nodes equal and it assumes no preferences between nodes. There is no effort to organize nodes or traffic in different categories and the network is assumed, by routing protocol, to be seated on a flat geometric plane. Hierarchical routing protocols give different roles to nodes. Some nodes are assigned special duties such as being a gateway. The notion of clustering comes to play. When a node wants to forward a packet to a destination and the the destination node is not in its neighborhood, it sends the packet to a special node called gateway. The gateway then sends the packet to other gateways until it reaches the gateway with a direct connection to the destination. In geographical routing, routing is based on the geographical positions of nodes. This means that a source node uses geographical address of the destination node to forward a packet instead of using its network address. Each node should be aware of its own geographical location and the destination's geographical location. Thus, in this type of protocol, routing can be done without the knowledge of the network topology and route discovery process.

2.3.1.2 Link-state, Distance-Vector routing protocols

In MANETs, there are families of routing protocols which inherit the characteristics of conventional routing protocols. These types of protocols utilize conventional routing concepts,

consider all the characteristics and limitations of MANETs and propose some novel techniques to deliver reliable services.

2.3.1.3 Proactive, Reactive, and Hybrid routing protocols

In proactive routing protocols, each node (router) proactively maintains a fresh list of routes to all other nodes in the network. These routes will be used by each node whenever they are needed. Because routes already have been found and reside in the routing table, no latency is imposed on the network prior to forwarding a packet to find the needed route. Routes are computed periodically through the dissemination of control messages throughout the network, which impose heavy traffic to the network. Examples of this kind of protocol are *OLSR*, *DSDV*, and *TBRPF* [41]. On the other hand, in reactive routing approach, the routing protocol does not take the initiative for finding a route to the destination until it is required, and there is no topology table in nodes. This type of protocol, which are also called *on-demand*, do not flood the network with periodically disseminated control traffic in order to maintain fresh route to all other nodes. On-demand protocols start the route discovery process whenever it is needed, and the route is maintained through the route maintenance procedure until the route is no longer required. Therefore, they experience the route discovery delay prior to forwarding a message to a new destination. When node S wants to transmit data to node D, and it has no valid route to D, it starts a route discovery process. In route discovery S starts broadcasting to all nodes searching for node D. The process is completed when the route is found or all possible route permutations have been examined. After the route has been found, the route maintenance procedure maintains the route until either the destination become inaccessible, or the route is no longer needed. The main advantage of this type of protocol is that the wireless channel does not need to carry a large amount of routing overhead for routes that are no longer used. *AODV*, *DSR*, and *TORA* are reactive routing protocols [41]. Hybrid routing protocols try to combine the advantages of reactive and proactive routing protocols. For example, the *Zone* routing protocol defines a zone around each node. Inside the zone, routing is done in a proactive manner and outside the zone routing is done reactively. Doing this, nodes inside the defined zone are immediately available, and to access further nodes the protocol will use reactive approach in an optimized way. It should be mentioned that hybrid routing inevitably combines the disadvantages of reactive and proactive protocols.

2.3.2 MANET routing protocols descriptions

This section describes two commonly used MANET routing protocols.

2.3.2.1 AODV Ad-hoc On-Demand Distance Vector

AODV is a reactive (On-Demand) distance vector routing protocol which can be used in large wireless ad-hoc networks [37]. AODV is a distance vector routing protocol and it inherits the concept of destination sequence number from DSDV to become loop-free. In on-demand routing protocols there is no need for periodic broadcasts of route advertisements, however on-demand acquiring of a route introduces the route acquisition latency to the network. In AODV, which uses symmetric links between neighboring nodes [37], nodes that are not involved in any data communication process, neither maintain any routing information nor exchange routing tables with their neighbors. In AODV, routing process is done in four phases: path discovery, route table management, path maintenance, and local connectivity management. When a node wants to communicate with another node and it does not have any route information to destination, it starts the route discovery process. In this phase the source node sends the special packet called Route Request (RREQ) to all its neighbors. If a neighbor can provide the route, it sends back a Route Reply (RREP) message to the source node. Otherwise, it forwards the RREQ to its neighbors. The *RREQ* is propagated through the network until it reaches the destination node. All intermediate nodes on the path from source to destination maintain some information in order to build forwarding and reversing path from the source to the destination node. Thus, in path discovery process, with help of intermediary nodes, the forwarding and reversing paths between source and destination node are setup. In route management phase, each node has some information associated with each route table entry, such as destination sequence number of that entry, request expiration time, active route timeout, and route caching timeout. Also, each entry in the routing table contains destination address, next hop, number of hops, sequence number for destination, active neighbor for this route, and expiration time for this entry. Using this information, each node maintains up-to-date active routes to different destinations. Broken links can be detected using periodic *hello messages*, link-layer acknowledgments, or detecting failing attempts to forward a packet to the next hop. If a link breaks, the node notifies all other nodes that were using that link by sending a special *RREP* message with hop count of ∞ and sequence number that is one greater than previously known sequence number. Upon receiving a broken link notification, if the link is still needed, the node starts the route discovery process with a destination sequence number of one greater than the previously

known sequence number. These are all being done in the path maintenance phase. In local connectivity management phase, if a node does not send any packets to its active neighbors during a time named *hello interval*, it broadcasts a special *RREP* called *hello messages*. A *hello message* contains the time to live of one and a list of all nodes that the sending node heard their hello messages; therefore, a receiving node does not rebroadcast it and can be sure of existence of bidirectional link between sender and itself. These phases are not necessarily sequential, as each node tries to create new route if it is needed and it tries to maintain its active route while it maintains its local connectivity to its neighbors.

2.3.2.2 OLSR - Optimized Link State Routing

OLSR is a proactive routing protocol [15] where each node proactively maintains a list of routes to all other nodes in the network. These routes are used by each node whenever needed. There is no latency prior to forwarding the message since the routes have already been found and reside in the routing table. OLSR is an optimized version of conventional link state routing for mobile ad hoc networks. *OLSR* enhances the inherited characteristics of link state routing by reducing the size and number of broadcast control messages in the network. *OLSR* does this by introducing the *Multi-point Relays (MPRs)*. Each node selects a subset of its neighbors as *multi-point relays*. *Multi-point relays* are selected based on the fact that all the two-hop neighbors are accessible through them. Each node in the network maintains a list of nodes, which have been selected as a multi-point relay in a table called *MPR selectors*. Each node periodically broadcasts its *MPR selector* table and all the available routes to other nodes. Therefore, in *OLSR*, the route is a sequence of hops through *the multi-point relays* from source to destination. *OLSR* functions in four phases. Each node periodically disseminates *hello messages* which contain the list of all one hop neighbors and their connection status, that can be directional, bidirectional, or *MPR*. These *hello messages* will be received by all one-hop neighbors, and these neighbors will not broadcast *hello messages* any further. By receiving a *hello message*, a node determines if it has been selected by its neighbors as *MPR*. Each node maintains a list of *MPR selectors*, and periodically broadcast this table using a special type of message called *topology control message (TC)*. *TC* messages will be broadcasted throughout the networks, and all the nodes in the network will build their routing tables based on their one-hop neighbors table, *MPR selector table*, and *TC messages* they have received. *OLSR* uses the concept of sequence number, which identifies the most recent information, to get rid of controlling in-order delivery of control messages. *OLSR* particularly is a good option for dense and large networks. *OLSR*

is designed to work in a completely distributed manner and there is no dependency on any central entity.

2.4 Policy-based management

Nowadays, systems in general are becoming more complex and decentralized. Social, financial, political, and information systems are getting larger and comprise a great number of entities. Managing this type of system is time consuming and tedious. Managers have to spend a lot of time trying to govern different entities, monitor their behaviours, detect abnormalities, and make important decisions to act upon them. Large systems need a huge amount of expensive administrative tasks which are critical to maintain the system in a healthy state. A large body of research has been dedicated to system management and finding ways to automating administrative tasks.

In complex distributed computer networks, different types of entities are involved and collaborate to deliver the main service of the networks. In these types of networks human, computers, and peripherals are all bounded to each other to deliver resources to the networks' users. Due to this entity diversity and different requested services from users, network management is a complex and arduous task. Network management tasks include *fault management*, *performance management*, and *security management* [3]. In order to automate these facets of networks management, researchers have proposed different solutions. One of these solutions is policy-based network management (PBNM).

In PBNM systems, administrators do not deal with the detailed configuration of entities in the network. Instead they define high-level requirements of the expected network behaviour. PBNM systems then configure all the settings in a way to be compliant to the requirements of the system. High-level requirements are injected to the PBNM in the form of a policy. Policies are building blocks of PBNM systems. Policy is information which can direct or modify the behaviour of a system [34]. D. Agrawal et al. define policy as externalized logic which can drive the behaviour of a system [1]. In order to have a secure manageable system Matt Bishop in *Computer Security Art and Science* [8] considers a computer system as a finite-state automaton with a set of transition functions that change states. He defined policy as a statement that divides states into secure and nonsecure groups. A secure system is a system which can neither start, nor enter in a nonsecure state.

Policies define either long-term requirements of a network, such as supported traffic and specific level of QoS, or how a network should respond to events such as detection of a

malicious user. PBNM systems have a automated feed back loop which is used to make a network responsive to changes without requiring any human action. The general concept of policy-based management systems is depicted in 2.1.

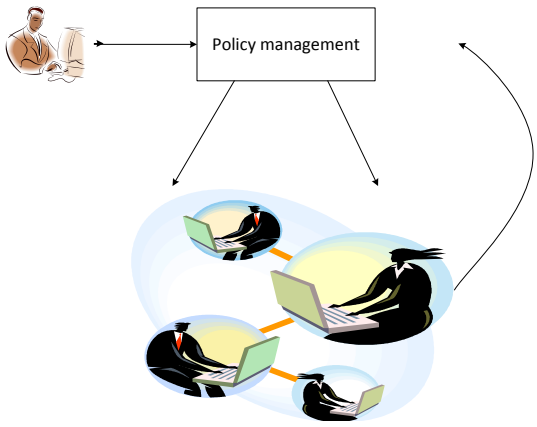

Figure 2.1: Policy-based Network Management

2.4.1 Policy-based network management architecture

In order to standardize the PBNM system concepts several working groups such as IETF [23] and DMTF [17] tried to propose an architecture for a PBNM system. A PBNM system as illustrated in 2.2 consists of *policy management tool*, *policy repository*, *policy decision point*, and *policy enforcement point*.

The policy management tool is used by administrators for defining and modifying policies that the system should implement in the network in order to keep the network's performance compliant to the requirements.

The policy repository is used to as a storage for the defined policy which can be accessed by other modules of the system.

The policy decision point (PDP) is responsible for interpreting the defined policies and instructing the network to take the proper action in different states of the system. The PDP ensures that the system is compliant to the policies all the time. The PDP receives inputs from the systems regarding the current state, and based on the defined policies directs

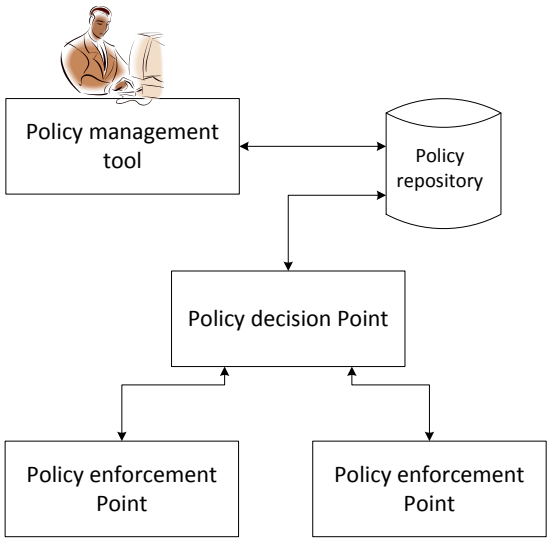

Figure 2.2: Policy-based Network Management Architecture

the system by either instructing an entity to take a specific action, or changing the system's configuration.

The policy enforcement point (PEP) is responsible for implementing and enforcing the PDP's decisions across the network.

There are two main challenges in designing and implementing policy-based management systems. The first one is how to translate the high-level policies, which are stemmed from requirements into a form which is interpretable with system components, and the second is how to extend the PBNM system to operate in a fully distribute environment.

To address the first challenge, different policy languages have been defined such as *Ponder*, *Ismene*, *Rei*, and *Policy Description Language (PDL)* [3]. Each of these languages take a different approaches to define policies. Some of these languages use structured programs developed specifically for expressing policies, some use formal logic languages, and others use set of simple structured rules in form of if ... then ... else [3].

To address the second challenge, Moris Sloman introduced networked resources as *managed entities* and placed them in *domains* [42]. The main promise of a *domain* was to group *managed entities (objects)* in order to apply management decision to them. *Domains* can include other *domains* and build a hierarchy. They then defined policies as objects which express a relationship among subject and target objects [42]. Having domain defined in the

managed environment, two approaches can be used to govern the policies in distributed fashion: *peer-to-peer*, *hierarchical*. In the former, each domain has its own fully independent network management system, and the coordination between network management system only happens if both participating parties benefit form this collaboration. In the latter, each domain has its own network management system but it is not independent of other network management systems, and they form a hierarchy which governs all the network entities [3].

Policy management has been extensively used in different aspect of network management, especially in security management areas such as *Role-based access control* and *Trust management*.

Notion of trust has been used in different areas such as sociology, economics, philosophy, and psychology. In computer science, concept of trust is used in different fields like autonomic computing and communication and networking. In communication and networking, Aivaloglou et al. defines trust as the quantified belief of a trustor regarding honesty, competence, security, and dependability of a trustee in a specific context. Also, trust can be define as the degree of a belief about the behaviour of other entities [13].

2.5 Wormhole attack

We are going to use the proposed framework of the thesis in a case study to detect wormhole attack in a *AODV*-powered MANET. In this section the wormhole attack is studied in more details.

2.5.1 How wormhole attack works

In this attack, attackers capture packets at one location in the network and forwards (tunnels) them to a distant location to be replayed in order to prevent benign nodes from having a valid image of the network's topology. The main goal of wormhole attack is to deceive benign nodes in a way that attackers can absorb high volume of network traffic to be passed through them. This attack can be done by one, two or more colluding nodes. Wormhole attacks can be launched using different techniques [4]. Sadeghi et al [39] has claimed that *AODV* is more vulnerable to a wormhole attack than *OLSR*. Sadeghi et al. [39] presents an analysis of the performance of *AODV* and *OLSR* powered MANET with and without wormhole attack. They have done their simulation in 1000m * 1000m area with 80 random waypoint mobile nodes. As *AODV* and *OLSR* are two dominant routing protocols MANET, there are lots of research in order to immunize *AODV* and *OLSR* powered MANETs against wormhole

attacks [4].

2.5.2 Wormhole attack launching techniques

Azer et al. [4] categorized the launching techniques of wormhole attack into five groups.

2.5.2.1 Out-of-Band Channel technique

In this technique, attacker nodes are directly connected, either by using long range directional wireless links or by direct wired link. As this technique needs special hardware, it is more expensive to launch and maintain. In this technique a colluding node tunnels received packets to the other colluding node residing in a distant part of the network using a direct link. Therefore, the packets will reach that part of the network faster and with fewer numbers of hops in comparison to packets which took the benign paths. Therefore, nodes residing close to wormhole ends choose the wormhole path to communicate with other nodes.

2.5.2.2 High Power Transmission technique

In this technique, any node can launch a wormhole attack. The malicious node uses high power transmission to broadcast received *RREQs*. Thus, its forwarded *RREQs* can reach other nodes faster compared to *RREQs* which are forwarded by benign nodes. This process increases the chance of malicious node to be on the routes established between a source and a destination.

2.5.2.3 Encapsulation technique

In this technique, when a colluding node hears a *RREQ*, it encapsulates it in a packet and sends this packet to another colluding node. The receiving colluding node will extract the encapsulated packet and replay it. In this way, all the nodes resided on the path between colluding nodes will be ignored in hop-count calculation process. Thus, the destination node will see the packet with very small hop count that will affect its path selection process. In this technique there is no need for any special equipment, such as high speed wire link or a high power source to create the wormhole. One way to prevent this type of wormhole attack is in the use of a routing protocol, which does not rely on the shortest path (minimum number of hops) like protocols which select fastest path. Motivation for this type of routing protocols is that a less congested longer route is better that congested shorter route [27].

2.5.2.4 Packet relay technique

In this technique, one or two malicious nodes can just play a role of an invisible bridge between two not-in-range benign nodes. The attacker just relays packets among two nodes, which are not in communication range of each other, without doing the mandated changes to the packet header and it deceives these two nodes that they are neighbours. After exchanging enough *hello messages* the wormhole will be established and the malicious node can take the control of the traffic among attacked nodes.

2.5.2.5 Using Protocol Deviations

Sometimes malicious nodes use protocol specification in order to disrupt the normal behaviour of the network. For example, a malicious node can ignore the back-off time enforced by *MAC* layer in order to win the competition of being the first to reach the destination.

2.5.3 Wormhole attack classification

Khabazizan et al. mention two modes of operation for wormhole attacks: *hidden* and *participation* mode [43]. In *hidden* mode colluding nodes do not use their identities to forward packets. Thus, they remain hidden to legitimate nodes. Basically in this mode, attacking nodes provide a virtual tunnel between two far-off legitimate nodes. Azer et al. named this type of wormhole as *closed wormhole attack* [4]. Well known wormhole detection schemes, such as *packet leashes*, *directed antenna* and *SECTOR*, deal with wormhole attack in this mode of operation. In *participation* mode, colluding nodes do not create a virtual link and they actively participate in routing as legitimate nodes and by sending packets through the wormhole they attract more traffic. Azer et al. divided these types of wormholes into *open* and *half-open* wormhole that respectively got their names based on the fact that both or only one of the colluding nodes reveal their identity to the routing protocol. After launching the wormhole and absorbing traffic, colluding nodes can launch DoS attacks by randomly dropping packets.

2.5.4 Wormhole Attack Impacts

We can see wormhole attack as a two-phase process. First, attacker should establish the wormhole tunnel and absorb network traffic. At this time - end of phase one - we can see the wormhole as a fast packet delivery service to network which cause the other nodes to not have a valid image of the network topology. In the second phase, wormhole has been already

used by other nodes to forward a significant amount of network traffic and attacker can monitor, edit, or selectively drop packets. Having control of the network traffic, the attacker has the opportunity to launch other type of attacks, like *DoS*. Periodically an attacker can switch an active wormhole off and impose a huge route rediscovery burden to the network. If a attacker decides to drop the network traffic, nodes close to a wormhole tunnel ends become sinkholes.

2.5.5 Wormhole attack countermeasures

There are two different approaches to deal with wormhole attacks: *proactive* (preventive) and *reactive* (detective). In proactive countermeasures usually special hardware is added to nodes. On the other hand, reactive approaches try to investigate some evidence that can prove existence of a wormhole attack. These two approaches define two lines of defence. In the literature, there are different categories of defence against wormhole attacks.

2.5.5.1 Location and time based solutions

In these solutions, nodes' locations and the time they send and receive packets are used in order to avoid using a wormhole to route packets. In these solutions, nodes are usually equipped by extra equipment, such as *GPS*, or they have tightly synchronized clocks.

Packet leashes: Hu et al. introduced the notion of *leash* where extra information is added to a packet in order to defend against wormhole attacks [22]. A *leash* is any information added to a packet in order to limit the distance a packet can be transmitted. Two types of leashes have been introduced: *geographical* and *temporal*. A *geographical* leash ensures that the distance between sender and receiver of a packet does not exceed a certain limit. A *temporal* leash ensures that a packet has an upper lifetime limit, and it can not be alive for more than a specific limit. In *geographical* leashes, nodes use loosely synchronized clocks in order to calculate the traveled distance of packet, and in *temporal* leashes, nodes use tightly synchronized clocks in order to calculate a precise life time for each packet and prevent a wormhole attack. Common authentication scheme or signature scheme like MAC and RSA can be used to ensure the authenticity of timestaps and location information in leashes. The main drawback of this method is the dependence on clock synchronization and need for extra equipment such as GPS devices.

Directional Antenna: Hu et al. [21] introduced *directional neighbor discovery*, *verified neighbour discovery*, and *strict neighbor discovery* protocols which use *directional antenna* to detect the existence of a wormhole attack in MANET. In these protocols directional information are shared between source and destination. Nodes use specific sectors of their antennas to communicate with each other. Therefore, a node receiving a message from its neighbour can calculate the orientation of the neighbour with respect to itself. This extra bit of information, makes wormhole discovery much easier than in networks with exclusively omni-directional antennas. This approach requires neither clock synchronization nor location information, and is energy efficient.

SECTOR: In secure tracking of node encounters (SECTOR) introduced by S. Capkun et al. [10], like packet leashes, the same principal of measuring distance between nodes has been followed. However, it only measures single hop distance using *Mutual Authentication with Distance-bounding* (MAD) protocol and a special hardware which is appended to nodes in order to make them able to rapidly exchange challenge nodes with each other. Using these bidirectional fast- challenge bit exchange, nodes estimate distance between themselves and their neighbours which lead to wormhole attack detection.

2.5.5.2 Recent approaches

Following we discuss some innovative methods to detect and prevent wormhole attack that they do not need any extra hardware.

Diffusion of innovations: Azer et al. [5] have suggested a preventive detective scheme based on a social science theory called *diffusion of innovations*. This mechanism can help nodes to detect or prevent wormhole attack even before any interaction with malicious nodes. They claimed that their scheme is totally decentralized and does not impose any computational complexity to the nodes. Diffusion is the process that innovation is communicated between entities in a social network. Diffusion of innovation theory tries to show how an innovation is adopted or rejected by members of a social network. The spread of a new idea is explained in this theory by defining five phases *knowledge*, *persuasion*, *decision*, *implementation*, and *confirmation* which members of the social network who can have different defined roles, should pass through. Members of network can belong to one of the five defined roles *innovators*, *early adapters*, *early majority*, *late majority*, or *laggards*. Based on evidences

in the network, a set of thresholds are defined and nodes compare their observations with these thresholds. If they find strong variations they become early adopters of the idea that a specific node is launching a wormhole attack. Otherwise, they just listen to their neighbours' opinions (roles) and adopt a role based on the rule which the adopted role will be less than neighbours' role by a degree [5]. Superimposing this actor's network on the network, authors claimed they can detect and prevent wormhole attacks in wireless networks.

Is detective?	Yes	Is preventive?	Yes
Used routing protocol	AODV	Is simulated?	OPNET
Needs routing protocol modification	Yes	Is mobility considered	No
Disadvantages	introduce end-to-end delay to the network	Advantages	distributed and does not add computational complexity to the nodes

Table 2.1: Diffusion of innovation summary

Path tracing algorithm: The path tracing algorithm is another wormhole detection mechanism dealing with networks which are using *DSR*, another on-demand routing protocol in which the routed packets contain the address of each device the packet will traverse [46], as their routing protocol. In this approach, neighbours communication has to be bidirectional. T. Sakthil et al. [40] introduced a two-phase process which can detect and prevent wormhole attacks. In the first phase, they calculate *prior per hop* distance, *per hop distance*, and *timstamp* which are placed in the *DSR* packet header during the route discovery process. In this approach, no tight clock synchronization is needed and each node uses its own clock. On a route, the distance between adjacent nodes is computed and then compared with consecutive neighbours distances on the same route, and if the difference exceeds a threshold the link becomes suspicious that there is a wormhole. Another parameter which is calculated is *frequent appearance* of a link in a path. *Frequent appearance* can be computed by dividing the number of times a link appears in a path by the number of all created links. Thus, if the difference of a link's hop distance field and the prior hop distance field is more than a threshold, and also the link's frequent appearance is more than a threshold, the link

is assumed to be a wormhole. In the second phase, if a wormhole has been detected, this information would be propagated through the network and the malicious node information will be added in *wormhole lists* in other nodes. Thus, a wormhole attack can be detected using information resided in *DSR* packets' header. In this paper, authors do not mention how they calculated the thresholds.

Is detective?	Yes	Is preventive?	Yes- it starts indetective mode
Used routing protocol	DSR	Is simulated?	NS-2
Needs routing protocol modification	Yes	Is mobility considered	Yes - Random way point
Disadvantages	Does not mention how to calculate the thresholds		

Table 2.2: Path tracing algorithm summary

WAP: Wormhole attack prevention algorithm [14], proposed a mechanism to detect the wormhole in a mobile ad hoc networks without utilizing any special hardware. Sun Choi et al. [14] suggest neighbour nodes monitoring method using a local timer called *WPT (wormhole prevention timer)*. This method is deployed to networks with *DSR* routing algorithm with disabled *DSR optimization* because it performs end-to-end signature authentication of routing packet and verification of sending RREP right . When a node sends a *RREQ* to the network, it starts a *WPT* and waits to overhear the rebroadcasted *RREQ* from its neighbours. Using these over-hearings, the node builds up a *neighbour node table* which records the sending and overhearing time of a specific *RREQ* from each neighbour. Assuming the speed of light for packet propagation, by calculating the round trip distance between the node and its neighbours, a node can use *WPT* to find out if one of its neighbours is located on the other side of a wormhole. In order to detect exposed wormhole attacks, the *wormhole route detection* mechanism is proposed, which calculates the delay per hop on each route, and compares it to *WPT*, and it should be smaller. Using this method, *WAP* can detect hidden exposed wormhole attacks. *WAP* places detected nodes related to a wormhole in a list called *wormhole* node list, and broadcasts it in the network.

Is detective?	Yes	Is preventive?	No
Used routing protocol	DSR	Is simulated?	Qualnet
Needs routing protocol modification	Yes	Is mobility considered	Yes - Random way point

Table 2.3: Wormhole attack prevention

Multi path routing protocols: Ning Song et.al tried to detect and locate wormhole attack in MANET with multi path routing protocols [43]. They suggested Statistical Analysis approach (SAM) for multi path routing networks. Multi path routing protocols such as SMR [29] detect multi paths to a destination and preserve them in a node. In case of route failure, nodes do not need to execute the route recovery process and they just use next available route. Multi-path routing protocols are claimed to have less management overhead than single-path routing algorithms such *AODV*.

In this paper, the authors tried to use local data gathered by each node to spot the wormhole attack. They calculated some statistics in normal environment, referred to as training environment, and use these statistics in the real scenarios where the network is under wormhole attack. They completely ignored mobility, and they discussed two types of topologies: clustered, and uniform. At the end, they claimed that their method, which does not need any extra hardware such as synchronized clock or GPS, can be used in intrusion detection systems.

Is detective?	Yes	Is preventive?	No
Used routing protocol	MR - derivative of SMR	Is simulated?	Not mentioned
Needs routing protocol modification	It needs multi-path routing protocol	Is mobility considered	No
Disadvantages	It depends on topology of the network and completely ignore mobility	Advantages	It brings vulnerability of multi-path routing algorithms into attention

Table 2.4: Multi path routing protocols

2.6 Research Gap

Mobile ad hoc networks are infrastructureless networks with no central management entity and frequently changing topology. These characteristics make them vulnerable to security attacks. Wormhole attack is a complex network layer attack, which tries to gain the control over the traffic flow in the network. In this chapter, several countermeasure against it have been studied. It maybe feasible to have a framework which requires neither extra hardware nor high computational power to detect and react upon the wormhole attack. Policies can be used to empower the framework toward detecting not only wormhole attack but also other type of abnormalities in the network.

Chapter 3

Proposed Framework

We propose a framework which can be implemented in MANETs' nodes in order to empower them to detect anomalies and act upon them. Our framework utilizes policies [1] to establish objective trust [44] among nodes. A node, equipped with the framework, is capable of determining if it can trust its neighbours not to be involved in launching an specific attack or selfish behaviour.

Throughout this Chapter we use a case study to illustrate the proposed framework. The case study assumes a set of nodes scattered within 1500 by 1500 square meter area. The nodes' mobility pattern is random way point [9]. Nodes are not clustered in different groups and there is neither a central administrative entity nor a connection to preexisting fixed network. Prior to introducing the framework's components we define several terms.

Anomalies: In this context, anomalies are deviations from normal behaviours which cause network resources to be wasted. Anomalies can be caused either by a malicious node attacking the network or by a node facing abnormal situations such as battery power shortage and act selfishly. [35].

Environment: This refers to information about geographical deployment environment, node's hardware and software characteristics, utilized protocols, and transmission media.

Healthy environment: An operational environment of a MANET without any anomalies such as an attack or resource shortage.

Unhealthy environment: An operational environment of a MANET with an active anomaly such as an attack. e.g. a MANET with hidden wormhole attack.

Resources in MANETs are scarce and valuable. Thus, efficient resource utilization in this type of network is critical [20]. Making a decision regarding the existence of a certain anomaly and properly defend against it in a completely distributed environment is a challenging task that our framework makes nodes in a MANET capable of.

Nodes have full TCP/IP protocol [28] stack implemented in them and they operate in a fully distributed environment with no central administrative entity. Nodes are on their own, and they are not directed by any means. They are not aware of possible attacks and anomalies [49]. Our framework first tries to educate nodes in MANET about these possible attacks and then, empowers each node to detect attacks and act upon them. By implementing our framework inside each node we are trying to make each node aware of possible attacks and how to react in case of encountering with them. We educate each node about a possible attack by means of files containing information about attacks and the proper ways to resist them.

For example, in our case study, we educate each node about the existing potential attack called wormhole attack [22].

3.1 How the framework works

A set of evidence is gathered by our framework. These evidences are defined by studying the network's *environment*, mission, and anomalies that the network is going to resist against. By studying network's *environment* and mission and throughly investigating the anomaly in *healthy* and *unhealthy* environment, we define two sets of policies: *investigation policies* and *executive policies*. Our framework uses these sets of policies with the gathered evidences to detect anomalies and act upon them. *Investigation policies* and *executive policies* are defined based on behaviours of the network in absence and presence of an abnormality. We inject *investigation* and *executive policies* into each node. Each node gathers information about the environment it is operating in, and uses *investigation policies* in order to make some assumptions about a possible source and cause of the anomaly. Each node periodically receives its neighbours' opinions about the possible anomaly. Based on its own assumptions and its neighbours' opinions, the node forms its opinion which will be shared by the neighbours. Based on the opinion, the node uses *executive policies* to react to the anomaly. Following the policy-based management philosophy, our ultimate goal is to reach the point that by only updating *investigation* and *executive* policies we can make the network more resistant

to variety of security attacks.

Formation and utilization of the framework's building blocks before and after deployment is depicted in figure 3.1.

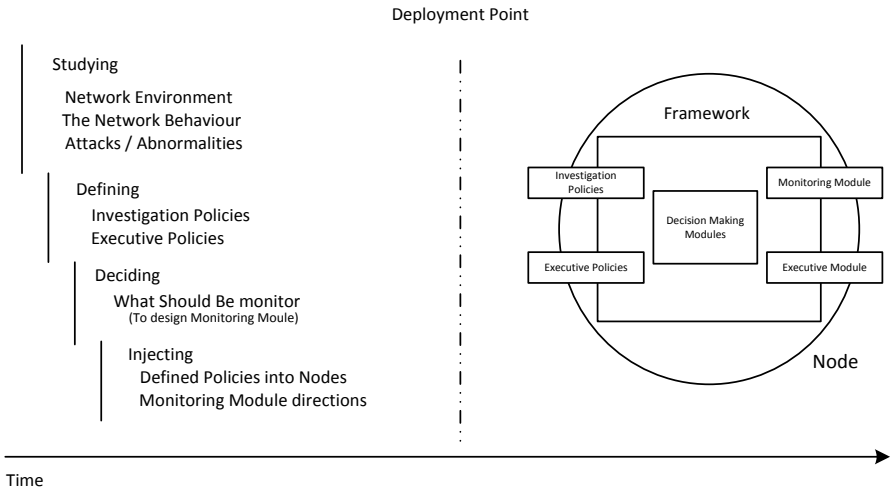

Figure 3.1: Framework before and after deployment

The framework allows nodes in a MANET to the point that they can form an opinion and take an action autonomously [26]. Decision making process needs prior knowledge, experience, and peer advice which is a very valuable asset in this process. In our framework, decision making is done by each individual node autonomously. Here we present two analogies to clarify the philosophy of our framework design.

Social Analogy: We can see a MANET as a society of autonomous entities [26]. In a society, although people are interacting with each other, decision making can be seen as an autonomic process in which each person based on his/her knowledge, experiences, observations, and received advices makes his/her decisions. We are using this analogy to design a framework to make each node capable of observing the environment and gathering evidences to make decisions in an autonomous fashion.

Vaccination Analogy: A vaccine is a biological preparation that improves immunity to a particular disease [48]. Policies in our framework work as vaccine agents that can stimulate a node to recognize and act upon a potential abnormality. Policies are injected into

nodes which are equipped with our framework and educate them about a particular disorder.

Therefore, by implementing our framework in each node we turn a node into a autonomous decision maker which is educated about possible attacks and is capable of reacting to them [26]. Figure 3.2 shows the decision making process carried out by each individual node.

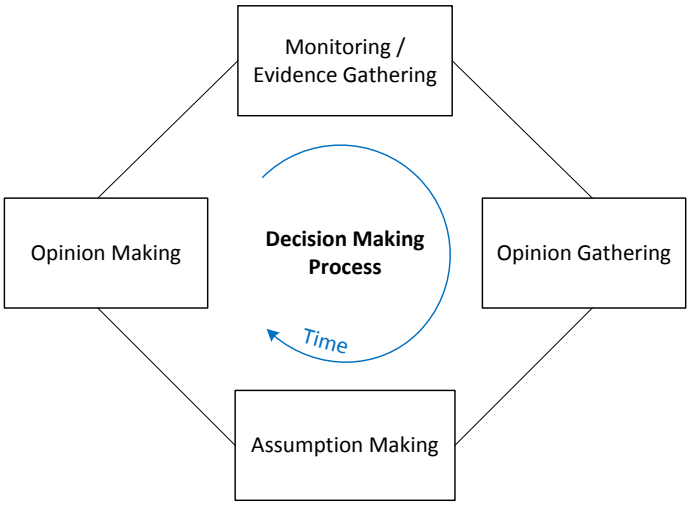

Figure 3.2: Decision making process

In the next section, we describe how to define policies based on the *environment* and possible abnormalities. In Section 3.3, we introduce the framework's components to be developed in each node.

3.2 Toward Defining Policies

MANETs are deployed in different environments utilizing a variety of routing protocols and pursuing different goals and because of nodes' mobility, their topologies are highly dynamic. Despite of all these varieties and dynamisms, based on network mission and deployment *environment*, prior to deploying a MANET, nodes' behaviours and network traffic patterns, in absence of malicious activities, to some extent are predictable. This predictability allows for policies to be designed and injected into nodes.

3.2.1 From Environment Characteristics to Policies

In this section, we introduce a method for defining sets of policies based on the network environment. *Environment* is the combination of geographical deployment environment, nodes' hardware and software characteristics, utilized protocols, and transmission media. A healthy environment is the environment in which the network in not under any attack and nodes are not functioning in stress mode. We try to immunize a network by injecting sets of policies into nodes in a network. Policies are defined by studying the *environment*, and network behaviour in *healthy* and *unhealthy* environment.

The predicted behaviours in *healthy* environment can be used as a touchstone for all active nodes to detect any deviation from the normal behaviours. In order to immunize the network against anomalies, we need to know normal behaviour of the network in *healthy* environment and network's behaviour while anomaly is active. Comparing these two sets of behaviours, we develop two sets of policies: *Investigation policies* and *Executive policies*.

Investigation policies are policies that are being used by each node to detect abnormalities and *executive* policies are policies that are being used by each node to perform proper reaction in response to detected anomalies. By defining *investigation* and *executive* policies we strengthen the immunity of the network against certain abnormalities.

3.2.2 Defining Policies

In order to define policies, we should closely study the physical environment, network mission, mobility pattern, nodes' hardware and software specifications, utilized protocols, and the normal and abnormal behaviour of the network in absence and presence of the anomaly. To do so, we introduce *Network specification table* in which all the different aspects of the *environment* is recorded and it can be used as a starting point to define policies. Table 3.1 illustrates a sample *Network specification table*.

Based on the *Network specification table*, we predict some behaviours of the network in a healthy state, which later will be used in defining *investigation* and *excutive* policies.

3.2.2.1 Studying the Network Behaviours

Based on the network characteristics and our goal in the specification table, we try to determine normal behaviours of the network in *healthy* and *unhealthy* states. Basically, we try to

Physical Environment	Describes the physical/geographical environment that the network is going to be deployed in
Network mission	Describes the network mission
Mobility Pattern	Based on the networks mission and type of nodes we can predict mobility pattern
Nodes hardware specification	Mention if nodes are equipped with a specific type of hardware or software
Utilized protocol	Describes the routing and other layers protocol
Our goals	Describes our goals in defining policies

Table 3.1: Network Specification Table

document the behaviours of the network in absence and presence of the anomaly. Based on this documented behaviours, we define our policies which will be injected to nodes. Figure 3.3 illustrates the policy defining process.

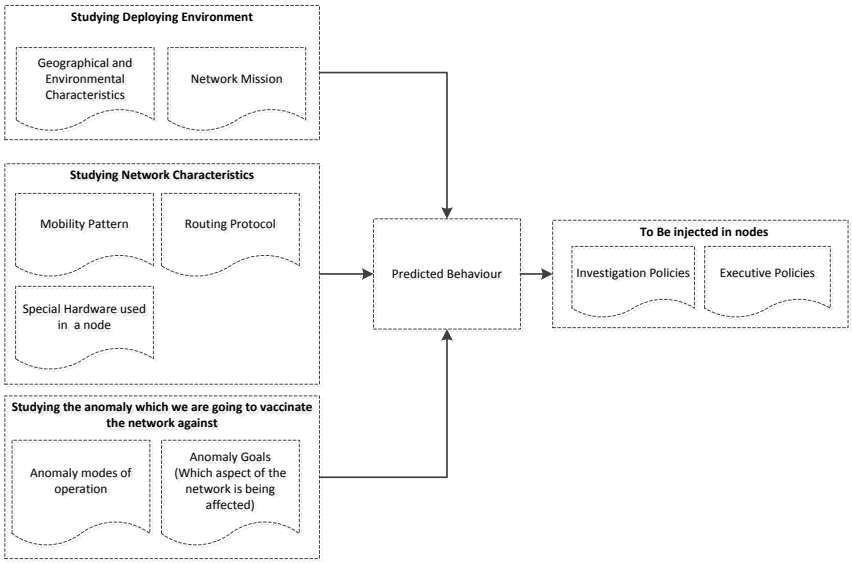

Figure 3.3: From Environment characteristics to policies

3.2.3 Defining policies for the case study

In the case study, first we define the *Network specification table*, then, we define several policies to be injected inside each node.

Physical Environment	The network is going to be used in a battle field in humid area in a forest
Network mission	Gathering topological information to be used in future infantry attacks
Mobility Pattern	Random way point
Nodes hardware specification	Nodes are not equipped with any GPS enabled devices. Nodes use 802.11. Nodes are mounted on wearables.
Utilized protocol	AODV [37]
Our goals	Making the network immune against wormhole attack

Table 3.2: Network Specification Table For the Case Study Scenario

Based on the *Network Specification Table* of our case study we can define several attributes, which can be used to represent a behavioural aspect. These attributes are defined for each of its neighbours. These attributes are used in the design of the monitor module and the definition of *investigation* and *executive* policies. Detailed information about wormhole attack can be found in chapter 2.

The least attached neighbour: An imaginary area with a radius of a transmission range of a node can be assumed around each node in a MANET. If the network is not very sparse, a node's neighbours can also be neighbours of each other. Based on the characteristics of exposed wormhole attack [4], one head of wormhole tunnel, as illustrated in Figure 3.4, should have the minimum number of common neighbours with the other neighbours. Thus, a neighbour with the most uncommon neighbours -neighbours which are not found in other received neighbour lists- can be malicious or affected by a malicious node.

The most traffic absorbent neighbour: The main intent of wormhole attack is to absorb a high volume of data and forwarding these data from one point in the network to another and then replaying them [22]. By listening to the traffic, we can find a node which absorbs the highest volume of traffic from other nodes. This can be an indication of existence of a wormhole tunnel in the network. Figure 3.5 illustrates a highly congested link created by a wormhole tunnel.

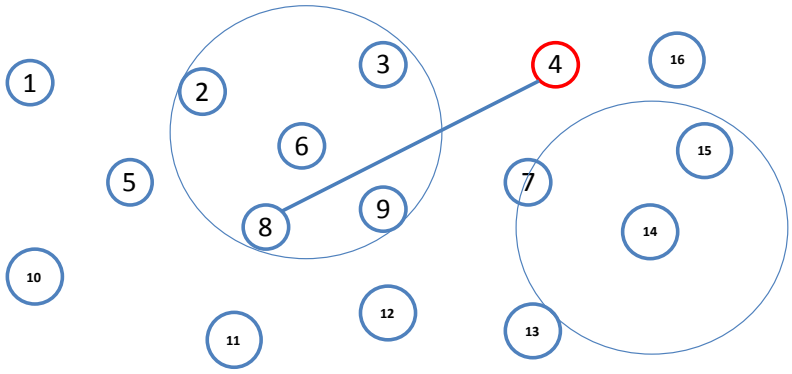

Figure 3.4: The Least Attached Neighbour

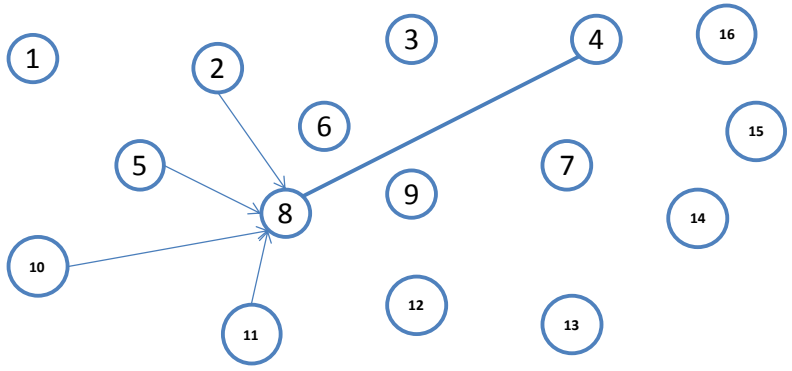

Figure 3.5: The Most Traffic Absorbent Neighbour

The furthest neighbour: There are different techniques used to estimate the distance between a node and its neighbours in a MANET [12]. In an exposed wormhole attack, usually the wormhole tunnel becomes congested [4] which causes communication delay. This would lead to the appearance of a node with a distance much larger than the mean of distances with other neighbours. As shown in Figure 3.6, the existence of a node with an unusually large distance can be the result of a wormhole attack.

The least mobile node: Due to the nature of wormhole attack, colluding nodes are usually connected to each other. This connection can be seen as a barrier for their mobility. Any deviation from the predicted mobility pattern and ratio of distance to signal strength can be used as an indication of malicious activities.

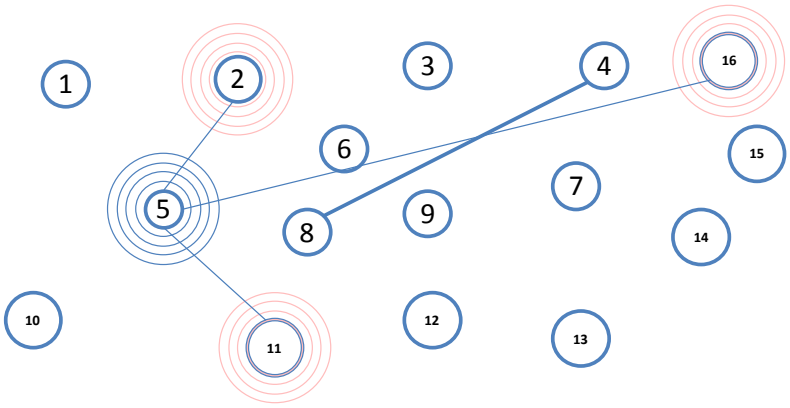

Figure 3.6: The Furthest Neighbour

Two very friendly nodes: When colluding nodes use encapsulation technique to form the wormhole [4], the network traffic between them increases abnormally. The high traffic volume between two nodes can be overheard and get detected by other nodes and be interpreted as sign of malicious activity.

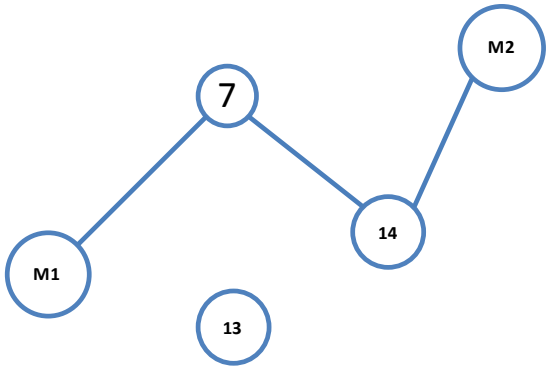

Figure 3.7: Two Very Friendly Nodes

According to the attributes, *investigation policies* can be defined as a set of neighbour ranking policies to rank the neighbours of a node based on the perceived likelihood that the node is malicious and may be launching a wormhole attack. For example, policies in the form of

```
If attribute(neighbor)==true then being_malicious(neighbour)++;
```

can be defined.Therefore, an *investigation policy* can be

```
If The_Least_Attached_Neighbor(B)==true then being_malicious(B)++;
```

and an *executive policy* based on the computed `being_malicious_factor` can be

```
If being_malicious_factor(B)>6 Then Ignore_Routing(B);
```

`Ignore_Routing` is a public method -which removes the routing information related to the malicious node from the routing table- in one of the framework components called *executive* which will be introduced in 3.3.3.

3.3 Framework components

In the previous section we introduced a method by which we can define sets of policies to be injected in nodes in order to educate a node about an anomaly. In this section, we introduce our framework which will be implemented in each node to make it capable of using the defined policies in order to immunize the node against an anomaly. Thus, each node can be immunized by leveraging the defined policies and the proposed framework.

By injecting the defined policies into nodes we vaccinate them against certain anomalies. We assume that nodes are mobile entities that are always interacting with the environment. Our proposed framework consists of three modules: *Evidence Gatherer*, *Opinion Manager*, and *Executive*. Figure 3.8 illustrates the proposed framework.

The framework is implemented in each node. As mentioned before, each node continuously interacts with the environment. Also, all the defined policies have been injected to it. Figure 3.9 shows the framework's components and the continuous interaction among a node and the *environment*. For better understanding, we superimpose the *policy defining process* to the diagram.

In the following sections, we describe the main modules of our framework and their duties. Prior to describing the framework's components, we define three terms: *Assumption*, *Opinion*, and *Introduced attacks* which will be used to describe the framework's components.

Assumption: As mentioned, the framework is implemented in each node. A node, based on gathered evidences and *investigation policies*, forms some assumptions about possible association between its neighbours and the introduced attacks. These *assumptions* are made based on the node's direct observations [44].

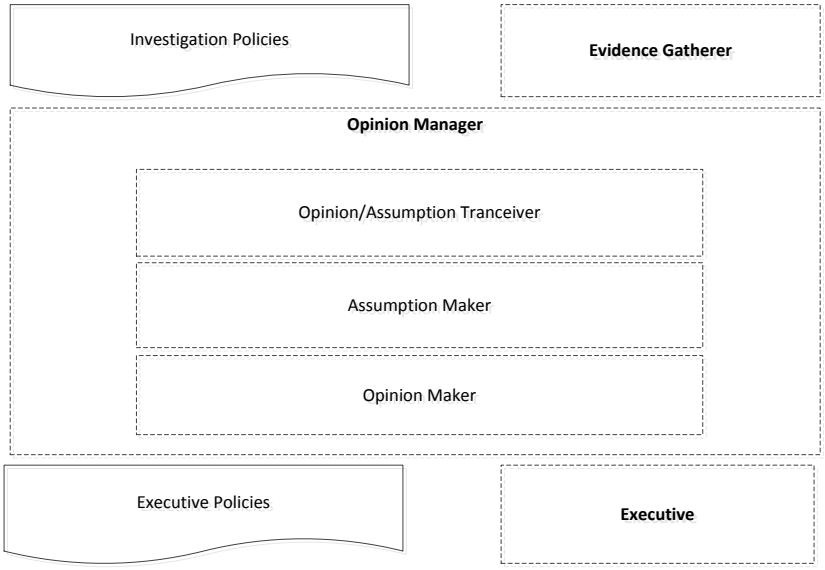

Figure 3.8: Framework (To be implemented in each node)

Opinion: Opinions are assumptions with higher certainty. In order to form an opinion about a relationship between one of neighbours and an introduced attack, a node not only relies on its assumptions, but also it takes into consideration its neighbours' opinions, which are received in certain intervals. Therefore, an *opinion* implicitly reflects internal *assumptions* and *external opinions*.

Introduced attacks: Abnormalities and the ways of reacting against them which are introduced to a node by means of *investigation* and *executive* policies are called *Introduced attacks*.

3.3.1 Evidence Gatherer

Evidence gatherer gathers evidences. It puts a node in promiscuous mode [20] and continuously listens to the network traffic and other important evidences such as signal strength, neighbours' hello messages, neighbours' exchanged topology control lists, and communication pattern among other nodes. Evidences are facts which can facilitate the decision making process for a node. For example, in a network which uses AODV [37] as its routing protocol, *evidence gatherer* would run some analytical algorithms to detect most active neighbour, most mobile neighbour etc.

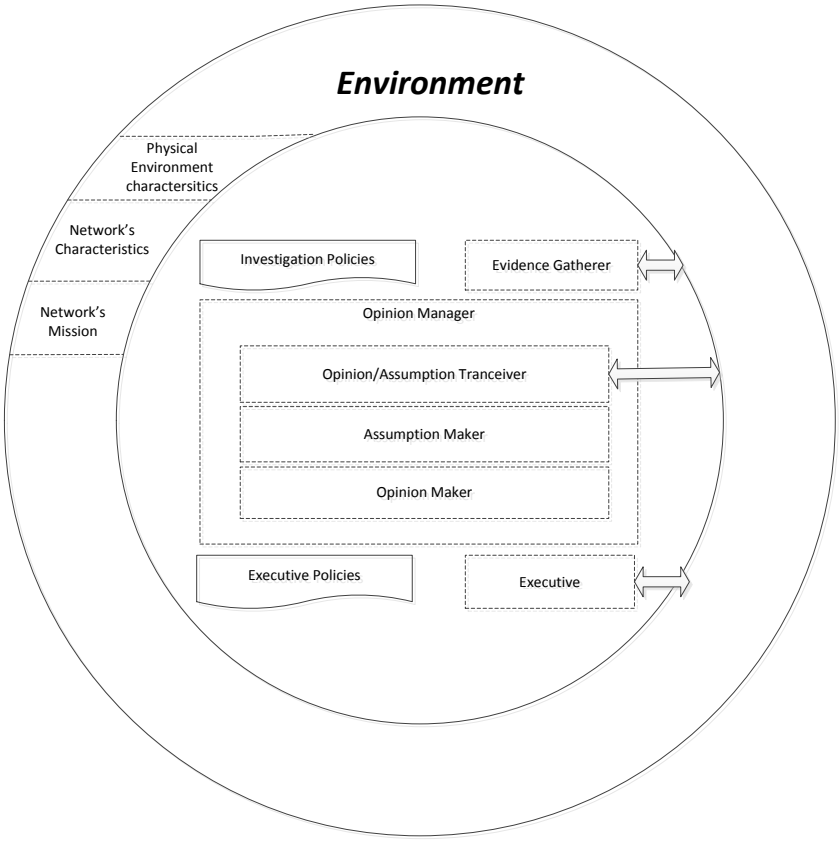

Figure 3.9: Framework interacting with environment

3.3.2 Opinion manager

The *Opinion Manager* is responsible for making decisions and direct the executive module to act upon the detected anomaly. It consists of three submodules: *Assumption maker*, *Assumption/Opinion transceiver*, and *Opinion maker*. Based on received evidences gathered by *Evidence gatherer*, it tries to find the source and cause of an *Introduced Attack*. While it is making assumptions about the source and cause of the anomaly, it also gathers its neighbours' opinions. By forming assumption and gathering other nodes' opinions, a node uses some statistical methods to make it's own opinion about association between a neighbour and an abnormality.

3.3.2.1 Assumption maker

Based on the evidences gathered by *evidence gatherer*, assumption maker is responsible for making assumptions about the association between neighbours and *introduced attacks*. *As-*

sumption maker's output is a list of nodes with maliciousness degrees regarding an *introduced attack* which will be exposed to *Opinion maker* module.

3.3.2.2 Assumption/Opinion transceiver

Assumption/opinion transceiver in a node is responsible to receive neighbours' opinions and transmits the node's opinion about an *introduced attack* to neighbours. This module exposes internal assumptions and external opinions to *opinion maker* module to be used in opinion making process.

3.3.2.3 Opinion maker

Opinion maker uses assumptions made by the *assumption maker* and opinions received by *Assumption/Opinion transceiver* in order to make opinions. Opinions express the association between a neighbour and introduced attack which is result of combining a node's assumptions and the node's neighbour's opinions.

3.3.3 Executive

Executive receives opinions from *opinion maker* and has access to executive policies. Based on the opinions and executive policies, *Executive* performs some reactive actions to prevent the malicious node from further disruption of the network performance. For example, if an opinion tells executive that node X is malicious to launch a wormhole attack, *executive* can ignore all routing messages to and from the malicious node.

3.4 The Framework in Action

In our framework, each node gathers evidences and other nodes' opinions about a potential attack. Based on the gathered evidences and a set of policies it makes an assumption about a potential attack. Having assumptions and other nodes' opinions, the node forms its own opinions. The formed opinions then are shared with other nodes and be used along with a set of policies to react to the potential attack. In this section, the overall behaviour of the framework is described in an example. Figure 3.10 illustrates the topology of MANET in which node M1 and M2 launch a half-open wormhole attack. Node M1 exposes itself to routing but M2 does not participate in routing mechanism and only sends received packets to the M1. Other nodes are benign nodes.

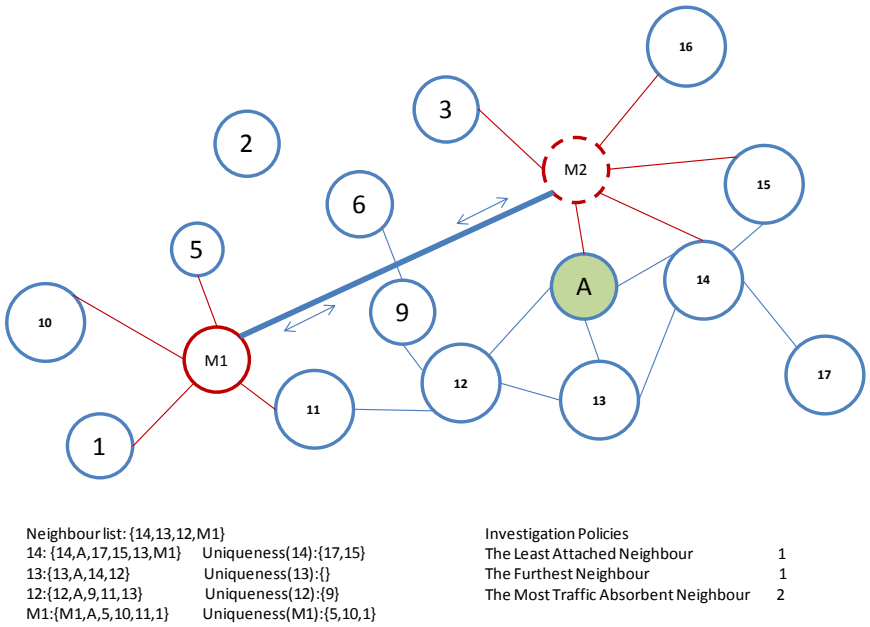

Figure 3.10: Example Topology

The process of opinion making for node A is described in the rest of this section. Upon joining the network, node A starts to gather evidences and other nodes opinions regarding a potential attack. Node A's evidence gatherer executes its *TheFurthestNeighbour*, *TheLeastAttachedNeighbour* and *TheMostTrafficAbsorbent* methods. Evidence gatherer listens to network communications and results of the three methods are recorded as evidences. Meanwhile, the internal assumption/external opinion module gathers other nodes' opinions embedded in received hello messages. Using *TheLeastAttachedNeighbour* method, nodes A and 14 detect M1 as the least attached neighour. Other neighbours of A detect either A or 14 as the least attached neighbour. This results show that the least attached neighbour method in nodes that are not directly connected to the wormhole can detect neighbours which are affected by a wormhole. *TheFurthestNeighbour* method in A estimates the distance between node A and its neighbours. A wormhole tunnel absorbs a large volume of traffic which can cause congestion and consequently impose delay on communication links. As it takes longer for node A to hear acknowledgment (response) hello message (see chapter 4) from M1, A detects M1 as the furthest neighbour. The same situation may happen for Node 14. Other neighbours of A could detect different nodes as their furthest neighbour. A operates in promiscuous mode and it can overhear its neighbour communications. *TheMostTrafiicAbosorbentNeighbour* method keeps track of communications and at a time it

can identify the node which communicates with a larger number of nodes. Malicious nodes which launch wormhole attack present themselves as good candidates for forwarding packets to destinations. Thus, in exposed wormhole attack *TheMostTrafficAbosorbentNeighbour* detects wormhole attack endpoints as the most traffic absorbent nodes. In case of hidden wormhole attack nodes which are affected the most by wormhole attack are detected as the most traffic absorbent neighbour. In the example, A and 14 detects M1 as the most traffic absorbent while other A's neighbours detect A and 14 as the most traffic absorbent neighbour. Results of these three methods are evidences which are used by the Assumption maker. The *assumption make*r uses investigation policies to quantify the gathered evidences in a way that it can quantify a potential attack. In investigation policies evidences are related to scalar values. For each neighbour, a variable called being malicious is maintained. In the example, at a given time, *assumption maker*, based on gathered evidences and investigation policies calculates values of being malicious of each neighbour for wormhole attack. These values represent the current assumptions. In the example, A's current assumption about wormhole attack would be M2 by value of 4. and other A's neighbour detects either A or 14 as the node with the highest being malicious value. A also, received other nodes opinions about wormhole attack which in the example topology 14's opinion of wormhole attack would be node M1 and other A's neighbours would detect A or 14 as wormhole attack endpoints. A receives other nodes' opinions. It forms an opinion based on its assumption and node 14s opinion that node M1 is a wormhole endpoint. Opinions are weighted arithmetic mean of assumptions and external opinions. Based on the opinion and investigation policies node A starts to react and boycott the M1.

3.5 Summary

In this chapter we have proposed a policy-based immunization framework for mobile ad hoc networks which can be implemented in each node to immunize it against certain abnormalities. The framework consists of several modules: *evidence gatherer*, *assumption maker*, *opinion maker*, *sssumption/Opinion transceiver*, *executive*. These modules turn a node into an autonomous entity that monitosr the network and looks for certain behaviours which are introduced to the node by xml files. Based on the observed neighbours' behaviours, the node forms an assumption about the possibility of a neighbour being malicious. Then, the node shares its opinion with its neighbours. Combining the observed behaviours with gathered opinions, the node is capable of making opinion with higher certainty.

Chapter 4

Implementation and Simulation

This chapter describes the implemented prototype of our proposed framework, and demonstrates the implementation of the proposed framework in the *AODV*-powerd MANET in order to prevent wormhole attack.

4.1 OPNET

There are several ways in order to validate a new framework or protocol in a networked environment such as: *mathematical modeling*, *simulation*, *hybrid* (which is combination of *simulation* and *mathematical modeling*), and *test-bed emulation* [33]. *Mathmatical modeling* is the fastest method, but when a complicated model with various factors is to be modeled, it is not accurate and it becomes inapplicable. *Simulation* models the interaction between modeling devices, and usually creates detailed packet-by-packet model for network activities. In order to compromise the significant amount of computational power and the time-consuming nature of *simulation*, sometimes *mathematical* modeling combined with *simulation* are used to model behavious of a network. This method is called *hybrid modeling*. *Test-bed emulation* is implementing a new framework or protocol in small scale on real devices. This method is more expensive and almost always involves unexpected engineering problems.

OPNET (OPtimized Network Engineering Tools) is the leading commercial discrete event simulator [33], which is highly used in industry and academia. OPNET follows object-oriented principals. A hierarchy of models are used in a network model in order to simulate network behaviour. In OPNET, network model contains *node models*, and *node models* consists of *processes*, *transmitters* and *receivers*. A *process model* simulates bahaviours of a node using a state transition diagram, in which transitions are conditions/events that occur

in a network's life span. The OPNET library contains many predefined network devices and protocols such as: routers, switches, fixed and mobile wireless workstations, etc. OPNET combines *C* language with state transition diagram, and offers a new language called *Porto-C* which is being used for designing and implementing process models. Also, C++ can be used to extend OPNET preexisting models. OPNET offers debugging facilities through *OPNET debugger (ODB)*, in which you can follow packets flow and movements of a mobile node in a simulated environment. In this chapter, OPNET is used for the simulation of the proposed framework. All the examples and sample codes are from a project which addresses the mentioned case study in Chapter 3.

4.2 Implementation considerations (prerequisites)

In OPNET, MANETs can be modeled using the *manet_station_adv* node model. Several predefined mobility patterns are offered by OPNET, and custom trajectory patterns can be implemented. In *manet_staiton_adv* the full *ip* stack is implemented. In the *ip layer*, the *ip* process has a child process called *manet_mgr* that itself contains different *process models* which implement the functionalities of *AODV*, *OLSR*, *TORA*, etc. The proposed framework is going to extend the *aodv_rte (AODV routing model)*. In order to utilize the proposed framework in the case study, wormhole attack should be simulated in OPNET. There is no predefined model that implements wormhole attack. In order to launch the wormhole attack, *manet_staiton_adv* process model has been extended to *manet_staiton_adv_with_wormhole* by adding one *process*, *WormholeTunnel*, and one set of point-to-point receiver and transmitter. Figure 4.1 illustrates the *manet_staiton_adv_with_wormhole* node model.

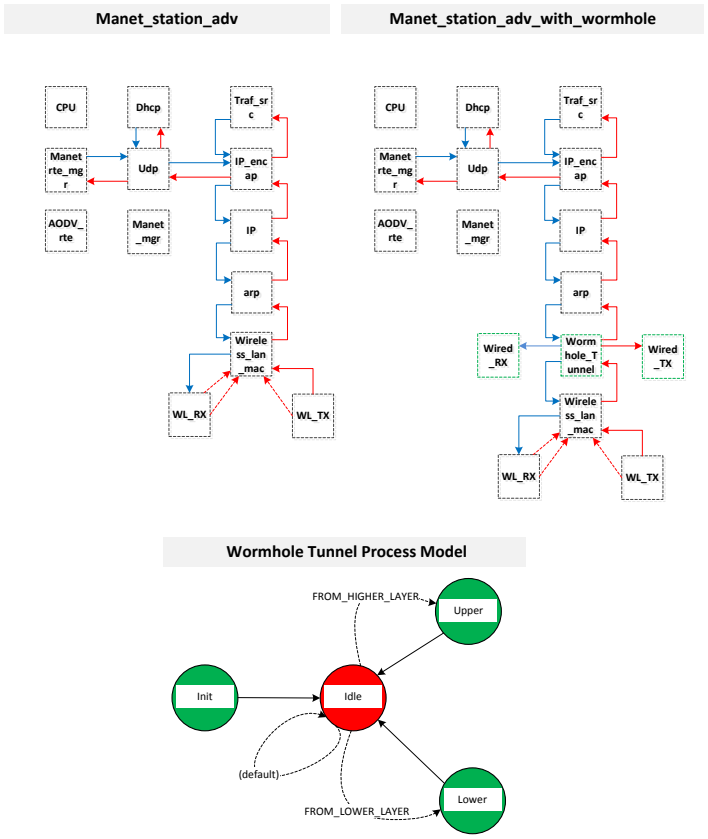

Figure 4.1: Manet Station Advanced With Wormhole node model

In order to turn each MANET node to an active traffic monitoring module, which is needed by the *evidence gatherer* of the proposed framework, each node should be capable of performing in promiscuous mode [49]. To do so, *manet_staiton_adv* is extended to *manet_staiton_adv_promiscuous* which enables a node to operate in promiscuous mode.

4.3 Implementation of framework components

For the sake of the implementation and moving toward the policy-based immunization framework, we placed the framework's components in a layered architecture. Figure 4.2 shows the framework's components from an implementation perspective.

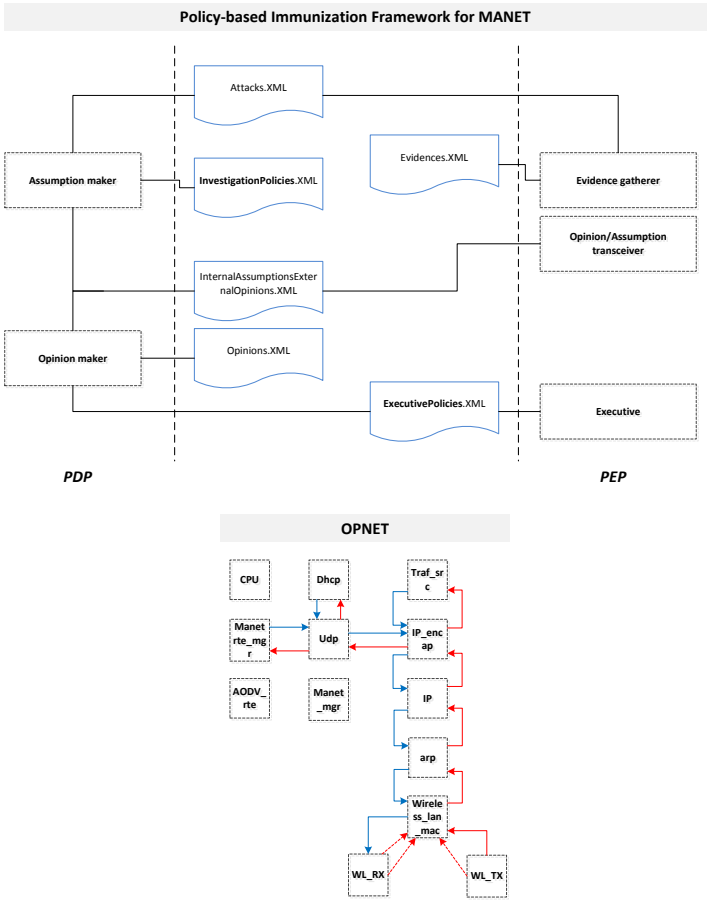

Figure 4.2: Framework From Implementation Perspective

In our framework, each node is a *PDP* and a *PEP*. Framework components are defined as classes in *aodv_rte* process model. In each node, in the *Begin_sim* interrupt, objects of these classes are declared and their operations are started.

4.3.1 Evidence gatherer class

Evidence gatherer consists of public methods which can be called by the objects. There is an entry in *evidence.xml* for each method, which stores the results of execution of the method. When the object of this class is initiated, communications on the network are monitored, and methods are executed in intervals. *Evidence gatherer* records the *neighbours list* of all the neighbours and information about communications among neighbors. Methods are designed based on the values of attributes that a node assigns to its neighbours, such as attributes discussed in 3.2.3. For example, *TheMostTrafficAbsorbent* method, in each interval, finds the *most traffic absorbent* neighbor and saves the result into *evidences.xml*.

In the mentioned case study, we implemented three methods in *evidence gatherer*: *The least attached neighbour*, *The furthest neighbour*, and *The most traffic absorbent neighbour*.

The least attached neighbour: In AODV, a hello message is a route reply broadcasted with a TTL of one. A hello message is received by all one hop neighbours. To support a node being capable of finding the least attached neighbour, we add two fields to AODVs hello message structure: *neighbour list* and *neighbours' neighbour list*. When a node sends a *hello message*, it includes all its neighbours and its neighbours' neighbors to the body of the message. Thus nodes by exchanging *hello message* can have a view of the network topology within two hops. A node, for each neighbour A forms a set called *AllNeighbours(A)*. This set contains all neighbours of A. The set-theoretic difference of the union of all the *AllNeighbours* of other neighbours with respect to *AllNeighbours(A)* forms a set called *UniqueNeighbours(A)*. *UniqueNeighbours(A)* contains nodes which are only neighbours with A. The cardinality of *UniqueNeighbours(A)* is calculated as *Uniqueness(A)*. The neighbour that has the *Uniqueness* set with largest cardinality is selected as the least attached neighbour. If two or more neighbours have the same *Uniqueness* value, the one with larger number of neighbours is selected as the least attached neighbour. For example, for a node C its neighbour list and neighbours' neighbour list are as follows:

Neighbour List : {A, B, F, Z}
Neighbours' Neighbour list :
A : {A, B, C, D, H}
B : {A, B, C, D, F, H }
F : {B, C, F, H, W, Z}
Z : {C, H, Z}

The *UniqueNeighbours(A)*, *UniqueNeighbours(B)*, and *UniqueNeighbours(Z)* are empty sets but *UniqueNeighbours(F)* is {W}. Thus, F is the least attached neighbour.

The furthest neighbour: In AODV, local connectivity between neighbours is handled by *hello messages*. The current sequence number and IP address is included in each *hello message*. In our framework, in order to enable a node capable of calculating the distance between itself and its neighbour, we added two fields to the hello message structure: *stamp1* and *stamp2*. When a node A wants to send a hello message it generates a unique code that can identify the message and inserts the code into *stamp1*, and sets *stamp2* to null. When a

node B receives the hello message it checks the stamps. If *stamp1* is not null and *stamp2* is null, B has to reply to the received hello message. It generates a new *hello message*, copies the value of *stamp1* in the received message into *stamp2* of the new message and broadcasts the message to its one-hop neighbours. In this example the *hello message* generated by node B functions as an acknowledgement to the hello message generated by A. When A receives the *hello message* generated by B, it determines that the received hello message is an acknowledgement to one of its *hello messages* sent earlier. Using the stamps, recorded sending and receiving times, and the speed of light, A can estimate distances between itself and its neighbours [22]. The neighbour with the largest distance is selected as the furthest neighbour.

The most traffic absorbent neighbour: In our framework, nodes not only keep track of active routes, but also utilize the overheard communication between neighbours to gain a better insight of the network. For each neighbour, the number of nodes communicating with that neighbour is maintained. Nodes operate in promiscuous mode and thus a node can overhear all the communication happening in their vicinity - an area in which a node can receive signals. Each node maintains a list which records node pairs which are communicating with each other. Each entry in the routing table contains a destination address and a next hop address through which the destination can be accessed. A neighbour with the largest number of communicating nodes is selected as the most traffic absorbent neighbour. If two neighbours have the same number of communicating nodes, the one that appears the most in the next hop fields of the routing table is selected as the most traffic absorbent neighbour.

4.3.2 Assumption maker class

Assumption maker uses instructions in the *attacks.xml*, *investigation policies* in *InvestigationPolicies.xml*, and evidences gathered in *Evidence.xml* to form an assumption about the relation between a specific attack, which is introduced to system by *attacks.xml*, and a neighbour.

4.3.3 Opinion/Assumption transceiver class

4.3.4 Opinion maker class

Based on the instructions in *Attacks.xml* and the assumption made by *assumption maker* and received opinions from *Opinion/Assumption transceiver*, which are stored in *InternalAs-*

sumptionsExternalOpinions.xml, an *opinion maker* class makes an opinion about a possible relation between a neighbor and an introduced attack, and stores the opinion into *Opinions.xml*.

4.3.5 Executive

Executive utilizes *executivePolicies.xml* and opinions stored in *opinions.xml* to instruct the node to respond to potential attacks. *Executive* consists of methods which can perform actions such as removing a route from routing table and ignoring the incoming packet from a neighbor. In the case study, each attribute is quantified and related to neighbors, and in this way the final opinion regarding the possible relation between a neighbour and abnormality, based on the defined policies, is formed.

Collaboration among the framework's components is facilitated by a set of XML files as introduced by the name of *anomaly introductory files*. The XML files are injected to each node prior to launch This makes nodes ready to deal with abnormalities. Attacks are introduced to nodes using an XML file called *attacks.xml*. Each attack in *attacks.mxl* has its own signature, which is used in other XML files to refer to this specific attack. *Investigation* and *Executive* policies are placed in each node by *investigationPolicies.xml* and *executionPolicies.xml*. Nodes' assumptions and their neighbours' opinions are stored in *internalAssumptionExternalOpinions.xml*. Finally, nodes' opinions reside in *opinions.xml*.

4.3.6 Attacks.XML

The *attacks.xml* file stores the attacks that a node should be aware of and act upon. *attacks.xml* is the place that different attacks are introduced. Each attack has a signature which will be used by the framework's components to refer to this attack. *Assumption* and *Opinion* makers use this file to manage the policy checking, assumption and opinion making processes. Each attack has a tag. Attributes of a tag include:

signature: This defines the signature of an attack which will be used as a reference tag in other XML files. XML files and the framework's components refer to this attack using this signature.

investigationPolicyCheckingInterval: This defines the time interval in which *assumption maker* should check *investigationPolicies* against current *evidences* and make its assumption regarding a potential relation among the attack and neighbours.

executionPolicyChceckingInterval: This defines the time interval in which *Executive* checks the *ExectionPolicies* for the attack.

opinionMakingInterval: This defines the time interval in which *opinion maker* should check *internalAssumptionsExternalOpinions.xml*, make an opinion about the potential attack and store it in *opinions.xml*.

attacks.xml

```
<?xml version="1.0" encoding="utf-8"?>
<Attacks>
  <Wormhole
          signature="WA"
          investigationPolicyCheckingInterval="30S"
          executivePolicyChceckingInterval="30S"
          opinionMakingInterval="30S"
          opinionTarget="Neighbour"
          opinionValue="Scalar"
          opinionDissiminatingInterval="30S"
          opinionDissiminationScope="AllNeghibours"
          internalAssumptionWeight="2"
          externalOpinionWeight="1"/>
</Attacks>
```

Figure 4.3: attacks.xml

opinionTarget: This defines the opinions' target. In the case study, for wormhole attack assumptions and opinions are made for neighbours. The opinion's values are going to be assigned to neighbours.

opinionValue: This defines the node's opinion value type about the *OpinionTarget*. For example, the node's opinion about the wormhole attack is *neighbour A* by value of 5.

opinionDissiminatingInterval: This defines the time interval in which *OpinionTranceiver* sends out current opinions.

opinionDissiminatioScope: This defines how far an opinion can be disseminated. e.g one-hop neighbours, two-hop neighbours, or broadcast.

internalAssumptionWeight/ externalOpinionWeight: Opinions are made based on internal *Assumptions* and external *Opinions*. In making opinion process, internal *Assumptions* and external *Opinions* can have an unequal weight. In an attack, it is possible that one has the higher impact on the final opinion than the other.

4.3.7 Evidences.XML

Evidences are the network monitoring results that are exposed by *Evidence gatherer*. Entries in this file are public methods in *Evidence Gatherer*, which can be called from other modules. This file automatically updated by *Evidence Gatherer*, and contains captured evidences results.

sourceAssembly: This is the *Evidence Gatherer* module. In this modules, all the methods which are responsible for monitoring the network and gathering the evidences are defined. These methods are executed periodically based on different policies and gather evidences which are going to be stored in *evidences.xml*. Each evidence has its own attributes. For example, in the above *evidences.xml*, there is an evidence entry which has the following attributes.

method: This is the name of the public methods in*sourceAssembly*, which is used to calculate the value of this evidence.

returnType: This can be a scalar value or a node or a list of nodes.

description: This is a short description of this evidence.

Evidence tags have two sub-tags, *Value* and *Last RecordedTime*, which contain the last returned value, and execution date and time of the method.

4.3.8 InvestigationPolicies.XML

Assumption maker uses *Investigation policies* in order to make assumptions. As mentioned in section 3.2.2, *Investigation policies* are defined by close examination of the network's environment, behaviours, and mission. *Assumption maker* checks the gathered evidences, which are resided in *evidences.xml*, against *Investigation policies* to form an opinion about the potential attacks introduces in *attacks.xml*.

The main entries in this XML files are policies. Policies are grouped by tags with an attack signature from *attacks.xml*

Each policy set is associated with one attack. Policies try to facilitate the assumption making process by quantifying the observed evidences into the measurable entities.

checkingScope: This defines which entities in the system should be compliant with this set of policy. e.g *AllNeighbours* or *AllActiveNeighbours*.

indicatorVector: This is the variable or collection name which is used by *Assumption maker* to store the values representing the quantified evidences for the attack.

resultType: This defines the return value type of checking process of this policy set for entities in the *checkingDomain*.

assumptionFunction: In order to form an assumption to be stored in *internalAssumptionExternalOpinion.xml*, an aggregate function is needed to be applied to values calculated in assumption making process.

Policy sets are enclosed in *Policy* tags. Every policy consists of a *condition* and *action*. *Condition* is checked against all the entities in *checkingScope*, and in case of true outcome, the *action* would occur.

4.3.9 InternalAssumptionsExternalOpinions.XML

Assumptions, which are made by *Assumption Maker* and neighbours' opinions regarding a specific attack, are stored in this file. This file is used by *Opinion maker* to make current opinion about the attack based on *internalAssumptionWeight* and *externalOpinionWeight* in *attacks.xml*.

Internal *assumptions* and external *opinions* are being enclosed in attack-signature tags.

type: This determines a tag is an internal *assumption* or an external *opinion*.

time: This is the recorded time of this assumption or opinion.

Every opinion or assumption encloses the *Node* and *Value* which are presenting the quantified value of an assumption or idea at the *time*.

4.3.10 Opinions.XML

Opinion maker, based on the definition of an attack in *attacks.xml* and the assumption and opinions stored in *internalAssuptionsExternalOpinions.xml*, makes an opinion. The opinion is stored in *opinion.xml*.

Each opinion encloses *Node*, *Value*, and *contriutors* tags which represent the opinion about an attack at *time* and number of neighbours which contributed in forming this opinion.

4.3.11 ExecutivePolicies.XML

Executive policies are used by *Executive* in order to direct the node to act based on the current opinions.

Executive checks *executionPolicies.xml* against current opinions in each *executivesPolicyChceckingInterval*, which is defined in *attacks.xml*. Based on them, it executes one of its public methods, which consequently directs the behaviour of the node.

Evidences.XML

```
<?xml version="1.0" encoding="utf-8"?>
<Evidences sourceAssembly="evidenceGatherer.cpp">
  <TheLeastAttachedNeighbours method="TheLeastAttachedNeighbour"
                              returnType="Node" description="This method
                              returns a neighbour with the least number of
                              common neighbours in comparison with
                              other neighbours">
    <TheLeastAttachedNEighbour>
    <Value>
      Neighbour1 (192.168.0.1)
    </Value>
    <LastRecordedTime>
      20130530 11:48AM
    </LastRecordedTime>
    </TheLeastAttachedNEighbour>
  </TheLeastAttachedNeighbours>
  <Evidence>

  </Evidence>
</Evidences>
```

Figure 4.4: evidences.xml

InvestigationPolicies.XML

```
<?xml version="1.0" encoding="utf-8"?>
<InvestigationPolicies>
  <WA  checkingDomain="AllNeighbours" indicatorVector="WA_BeingMaliciou"
       resultType="Scalar"  assumptionFunction="MAX">
    <Policy>
      <Condition>TheLeastAttachedNeighbour</Condition>
      <Action> 2 </Action>
    </Policy>
  </WA>
</InvestigationPolicies>
```

Figure 4.5: investigationPolicies.xml

InternalAssumptionsExternalOpinions.XML

```
<?xml version="1.0" encoding="utf-8"?>
<InternalAssumptionsExternalOpinions>
  <WA type="Internal" time="20130528 11:10AM">
    <Node>Neighbour 3</Node>
    <Value>4</Value>
  </WA>
  <WA type="External" time="20130528 12:10AM">
    <Node>Neighbour 2</Node>
    <Value>7</Value>
  </WA>
</InternalAssumptionsExternalOpinions>
```

Figure 4.6: internalAssumptionsExternalOpinions.xml

Opinions.XML

```
<?xml version="1.0" encoding="utf-8"?>
<WA time="20130528 10:00AM">
  <node>Neighbour 3</node>
  <value>3</value>
  <contributors>3</contributors>
</WA>
```

Figure 4.7: Opinions.xml

ExecutionPolicies.XML

```
<?xml version="1.0" encoding="utf-8"?>
<ExecutionPolicies executerAssembely="Executive.cpp">
  <WA>
    <Policy>
      <Condition> opinions->value GT 6</Condition>
      <Action> IgnoreRouting(opinions->node)  </Action>
    </Policy>
  </WA>
</ExecutionPolicies>
```

Figure 4.8: ExecutivePolicies.xml

Chapter 5

Experimental Results

This Chapter presents an evaluation of the proposed framework using the case study described in Chapter 3. Section 5.1 describes simulations environment setup and different scenarios we used in our simulations. In Section 5.2, benchmarks for the evaluation of the proposed framework are introduced. Section 5.3 explains how we set up the framework in a way it can detect wormhole attack. Finally, Section 5.4 discusses the impact of using policies in detecting abnormality in a MANET.

5.1 Simulation Environment Setup

This work uses OPNET for simulations. The mobile ad hoc networks used have the characteristics presented in Table 5.1. Figure 5.1 demonstrates the initial topology of the network without any malicious nodes. In our environment we use AODV as it is one of the dominant routing protocols in mobile ad hoc networks. To enforce mobility, we used the *random way point* mobility pattern. The simulation runs for 600 seconds. We used *Out-of-Band Channel* technique to launch a wormhole attack. Colluding nodes are connected by a fixed duplex link. All wireless communications transceived between colluding nodes use this wired link. The launched wormhole attack is a half-closed wormhole attack. This means that only one of the end points of the wormhole tunnel is exposed to the routing process.

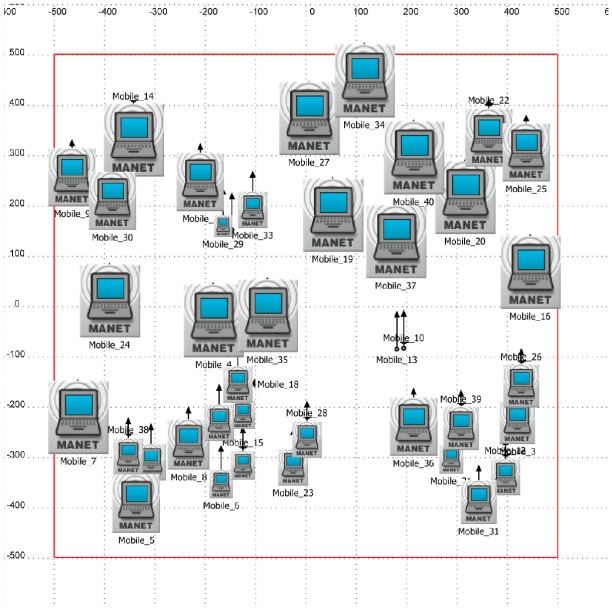

Figure 5.1: Manet Simulation setup 40 nodes in 1000*1000 m

Routing protocol	AODV
Percent of node moving	100%
AODV Hello message interval	1 sec.
Simulation time	600 sec
Mobility patern	Random way point
Seed value	Yes
Number of nodes	40 and 60
Node lacement of simulation area	Random
Packet Inter-arrival time	Exponential(1)
packet size	Exponential(1024)

Table 5.1: Simulation setup

5.2 Benchmarks

We compare the proposed framework with two benchmarks that represent two scenarios: *healthy environment* and *unhealthy environment*. The *healthy environment* has no malicious nodes while the *unhealthy environment* has malicious nodes that initiate a wormhole attack.

5.2.1 Healthy environment

We collected the average *hop count per route* and *MANET delay* in healthy state of the MANET. The *MANET delay* of a node is the average amount of time it takes to process a packet. Figure 5.2 and 5.3 compare these two metrics in the 40-node and 60-node scenarios. As expected, the *average hop count* is larger in 60-node scenario. Consequently, the *MANET delay* is higher in 60-node scenario. A successful wormhole attack should show a smaller average hop count than topology that does not have wormhole attack. The reason is that wormhole attacks are designed to convince benign nodes that the hop count is shorter than it really is. The MANET delay is longer since wormhole endpoints have to process more packets than a benign node, and it imposes queuing time to each packet being processed.

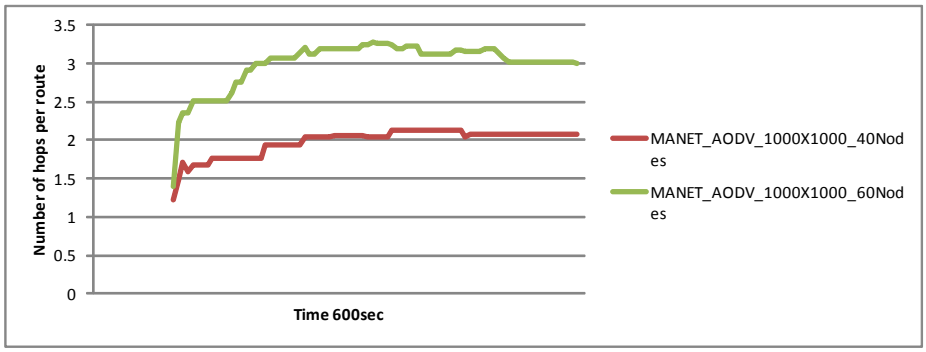

Figure 5.2: Average hop count per route in healthy environment

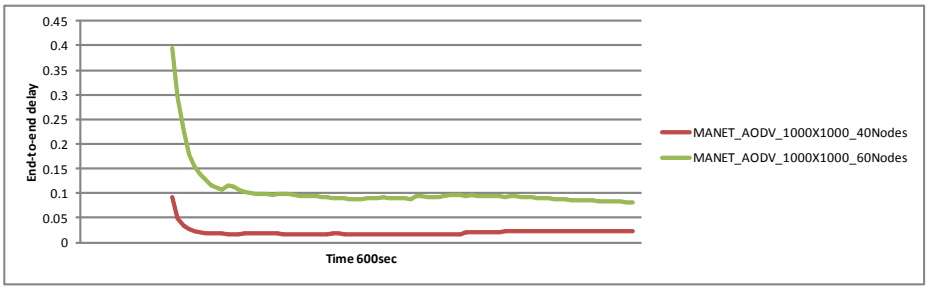

Figure 5.3: Average MANET delay in healthy environment

5.2.2 Unhealthy environment

The launched wormhole attack is a *half-closed* wormhole attack. Scenarios with 40 and 60 nodes without wormhole attack and with one and two wormhole attacks are simulated and

the results show that we have successfully launched the wormhole attack. Figures 5.4 and 5.5 demonstrate the average *hop count per route* in 40-node and 60-node scenarios respectively.

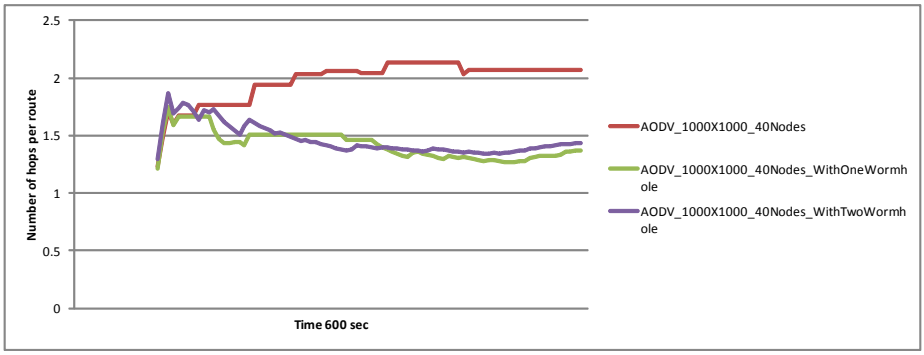

Figure 5.4: Average hop count per route 40 node scenario

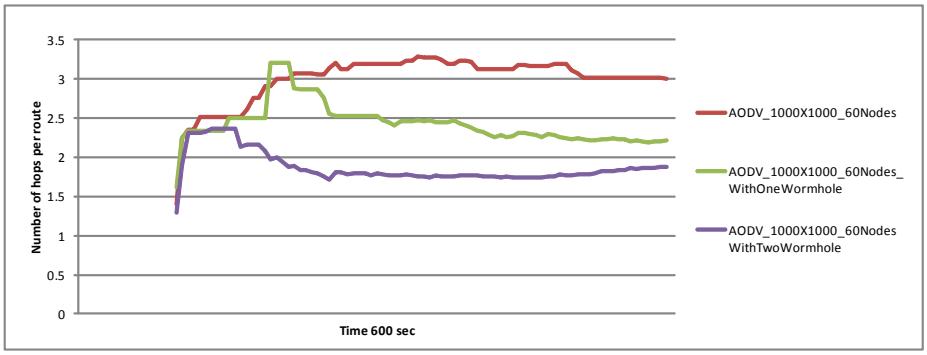

Figure 5.5: Average hop count per route 60 node scenario

As depicted in Figures 5.4 and 5.5, by launching wormhole attack, the average hop count per route is reduced.

We have also measured node's *MANET delay*. Simulation results are illustrated in Figures 5.6 and 5.7. The simulation results show the higher packet processing time imposed by the increased delay incurred by malicious nodes

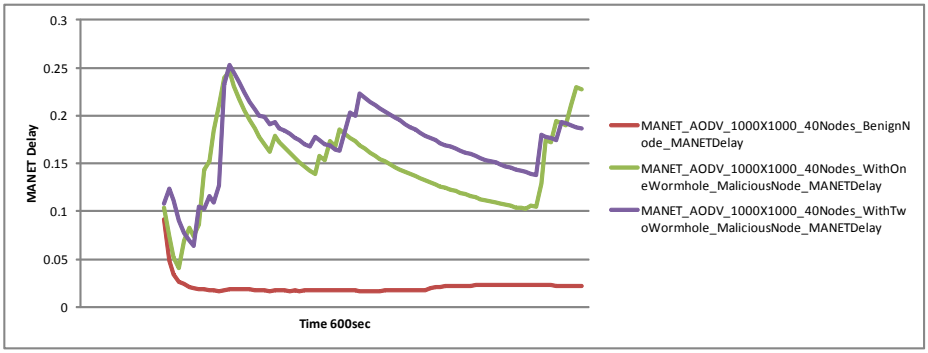

Figure 5.6: MANET delay of Malicious and benign nodes in 40 node scenario

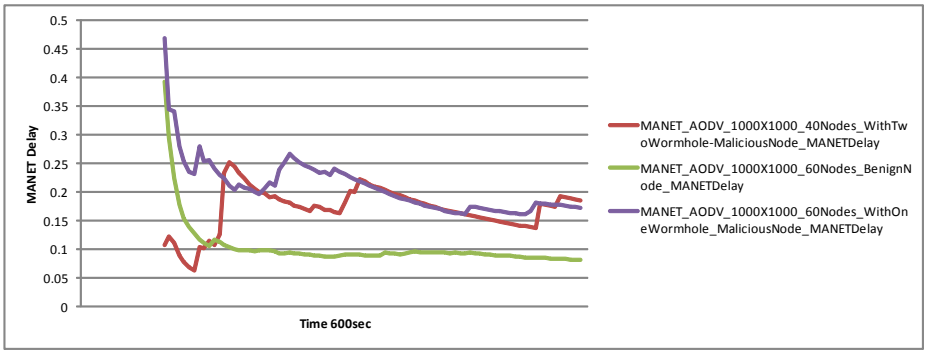

Figure 5.7: MANET delay of Malicious and benign nodes in 60 node scenario

5.3 Framework settings

The effectiveness of the framework relies on the values assigned to *assumption making interval*, *opinion making interval* and *opinion dissemination interval* (see Chapter 3 for more information). These are operational factors and they specify how often the framework should execute to detect specific attacks. These values are specified as a function of the interval between *Hello messages* (see chapter 2 for more information). Table 5.2 presents the values assigned to each of these factors during the experiments. There is a weight associated with an internal assumption and one associated with an external opinion. These are also specified in Table 5.2.

Simulation Runs	27
Intenal Assumption Weight	1
External Opinion Weight	0.75, 0.50, 0.25
Assumption Making Interval	1, 3, 5 * *Hello message Interval*
Opinion Making Interval	10, 15, 30 * *Hello message Interval*
Opinion Desimination Interval	10, 15, 30 * *Hello message Interval*

Table 5.2: Factors impacting performance

In order to quantify a neighbour's behaviour, each node assigns values to attributes that characterize the a specific aspect of behaviour. There are three attributes associated with a half-open wormhole attack: *The least attached neighbour*, *the most traffic absorbent neighbour*, and *the furthest neighbour*. Table 5.3 shows the value assigned to each of these attributes of a node's neighbour when it observes one of these behaviours. These values are used in the calculation of malicious behaviour (see chapter 3). In the experiment we used three combinations as presented in Table 5.3.

	Combination 1	Combination 2	Combination 3
The least attached neighbour	1	1	1
The Most Traffic Absorbent Neighbour	1	2	3
The Furthest Neighbour	1	1	1

Table 5.3: Quantifying setting combinations used in experiments

We defined four simulation scenarios: *forty node with one wormhole*, *forty node with two wormhole*, *sixty nodes with one wormhole*, *sixty nodes with two wormhole*. For each of these scenarios, we used all the operational settings presented in Table 5.2 and for each run we used all three combinations mentioned in Table 5.3. This means all the combinations of the values in Table 5.2 and 5.3 are used in all the scenarios.

5.3.1 Experimental results

This section presents the result of our experiments in evaluating different assignments of values to operational and quantifying settings.

We used *OPNET debugger - ODB* facilities in order to monitor the behaviour of nodes in the network. Using ODB, the *neighbours lists* and the *routing tables* are logged. Also, we have monitored our framework xml files to gain insight about the overall functionality of the network. For example, *opinions.xml* plays the role of a log file in investigating if a malicious node is detected by a benign node or not.

Table 5.4 presents the results of our experiments. The first row shows the average number of nodes interacting with colluding nodes during the simulation. The second row shows the average success rate of detecting the existence of wormhole attack by benign nodes, and the third row shows the highest success rate of the wormhole attack detection in all simulation runs.

Average number of node interacting with colluding nodes is calculated using OPNET debugger facilities. Nodes' *opinions* are studied and it was checked if during the simulation they identified malicious nodes. In each scenario, the average number of times nodes detect malicious nodes are calculated. This average is calculated for all the nodes which interacted with the malicious nodes during the simulation. For each scenario, simulations was run with all the combinations of values in tables 5.2 and 5.3 with no repetition.

Table 5.5 represents the configuration that provided the best average success rate of detecting the existence of a wormhole.

5.4 Impact of using the proposed framework

We use the configuration settings that gave us the best average success rate (see Table 5.5) to investigate the effectiveness of investigation and executive policies. The xml specification of the policies are in Appendix A. The policies used in the experiments are informally described bellow.

	40-Nodes one wormhole	40-Nodes Two wormhole	60-Nodes one wormhole	60-Nodes Two wormhole
Average Number of Node Interacting with Colluding Nodes	62%	80%	50%	78%
Average Success Rate of Detecting the Existance of wormhole	83%	82%	76%	80%
The Best Succes Rate of Detecting the Existance of wormhole	87%	88%	81%	83%

Table 5.4: Experiment results

Operational settings			
Internal assumption weight	1	External opinion weight	0.5
Assumption making interval	5 * *Hello Message Interval*	Opinion making interval	15 * *Hello message interval*
Quantifying settings			
The least attached neighbour	1	The Most Traffic Absorbent Neighbour	2
The Furthest Neighbour	1		

Table 5.5: Settings concluded from experiments to be used in wormhole attack resistant network

Name: Add to suspicious list
Condition: If value of an opinion about a node is equal or greater than 2.
Action: Add the node to the suspicious list.

Name: Suspend route
Condition: If the value of an opinion about a node is equal or larger than 3 and node is already in suspicious list.
Action: Suspend the routes in which the next hop is this node for 5 * CONTRIBUTORS * *opinionMakingInterval* and send RERR.

Name: New neighbour alert
Condition: If a new neighbour is added to the neighbours list.
Action: Send out a hello message immediately.

Name: Alter Being_malious value
Condition: If a node is the least attached neighbour.
Action: Add 1 to the being_malicious factor of the node.

An XML parser is designed in which all XML files can be interpreted and associated to methods and data structures in the system. This light-weight customized XML parser is designed because using policy languages such as *Ponder* [3] in MANET imposes high computational overhead to nodes.

Having policies injected in nodes, which are provided by the framework, we have run the 40-node and 60-node simulations. We measured the average *hop count* and *MANET delay*. Figures 5.8, 5.9 demonstrate the comparison between MANETs in which nodes are not equipped by the framework with MANETs equipped by the framework. The results show that policies which are tailored to detect wormhole attack are successful in returning the network to the healthy state. Also, successfully returning a MANET to the healthy state provides good evidence of the collaborative opinion making scheme of the framework.

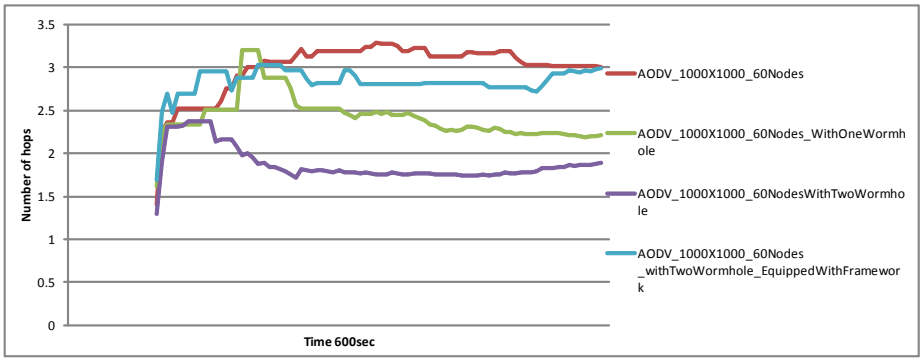

Figure 5.8: Average Hop count per route in MANET equipped with the framework

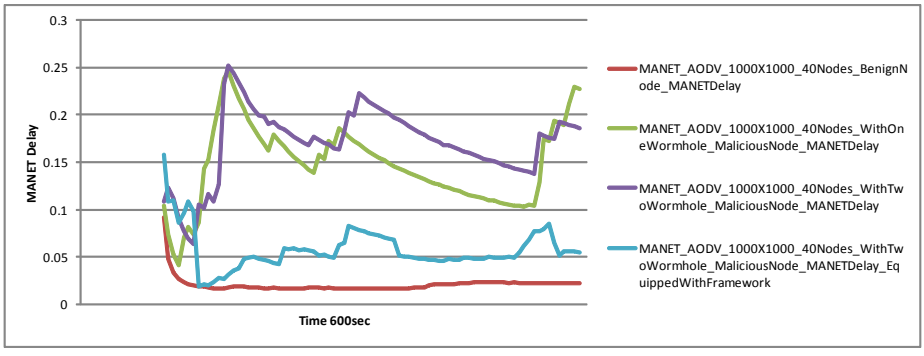

Figure 5.9: Average MANET Delay in MANET equipped with the framework

Chapter 6

Conclusions and Future works

This thesis relates to the area of mobile ad hoc network security, and its focus is on design and implementation of a policy-based immunization framework for securing MANETs. Section 6.1 presents the conclusion. Section 6.2 presents the contribution of this thesis, and 6.3 describes possible future work.

6.1 Conclusion

Our proposed policy-based immunization framework in used to turn each mobile node in a MANET to autonomous decision maker, which can reflect on possible abnormalities. Policies are used to guide decision making by nodes in a MANET. Policies can be designed to address a specific type of attack such as wormhole attack.

The proposed framework in flexible and extensible. Decision making regarding a specific attack and determining the proper reaction against it is guided by using policies defined in each node. Currently, we use a set of policies to detect wormhole attack but the framework is flexible enough that it is possible to introduce new type of policies that address other types of attacks. The framework also utilizes a collaborative voting scheme which mitigate the effect of the fact that a node can be physically captured by an attacker.

The proposed framework is modular. Communication among modules are facilitated by means of *xml* files. This kind of behaviour makes module loosely coupled, and another enhanced module(s) can be easily added to the framework.

We can successfully detect the existence of wormhole attack in a MANET without adding any extra hardware to nodes. Standard protocols are not modified, and only few fields are added to the local connectivity management packet called *hello messages*. There

is no need for any centralized entity to ensure the authenticity of communicated messages, and nodes became educated in a way that they can make more informed decisions.

We validated the functionality of our framework by implementing a case study and carrying out experminets for several scenarios using OPNET.

6.2 Contribution

By using our framework nodes become autonomous decision makers who can detect and act upon an abnormality more reflectively than reactively. Our framework is a trust management system which utilizes policies to help a MANET delivering secure and reliable services. We have used policies in order to direct components of a trust management system, evidence manager, opinion manager, and decision maker in more granular way than the current trust management systems. The proposed framework establishes trust regarding a specific abnormality between nodes. This approach helps a MANET to be more focused on potential abnormalities in different circumstances.

6.3 Future work

There is a good deal of room for enhancing either the current modules or adding new ones to the proposed framework. For example, a *validator module* can be added in order to validate final *opinions*.

The algorithms for quantifying the proposed attributes, which are the sign of potential abnormalities can be enhanced and become more complex to support more conditions.

We can define a trust level for each neighbour of a node. In that case, received opinions from neighbours regarding a specific attack can influence a node opinion differently. For example, a one-hop neighbour, whose opinion about an attack was close to a node opinion for the last three opinion-gathering-cycle, can be given a higher trust level than other one-hop neighbours with variant opinions.

A policy delivery mechanism can be defined in a way that in case of the development of new sets of policies they can be deployed to a MANET, which is operating on a field.

Appendix A

Policies

A node's *assumption maker* uses *investigation policies* in order to quantify its neighbours' behaviours. *Investigation policies* are grouped and tagged by an attack signature from *attacks.xml*. Policy set A.1 demonstrates the *investigation policies* created from the framework setting values in order to detect and react against wormhole attack.

Executive uses *executive policies* to act upon a detected attack. *ExecutivePolicies.xml* exposes public methods of *executive* class. *Executive* retrieves the latest opinion about a potential attack from *opinions.XML*. Based on the retrieved values, which can be the indication maliciousness of a neighbour, it executes a method from executive class. Policy set **??** illustrates the *ExecutivePolicies.XML* created from the framework setting values in order to detect and react against wormhole attack.

Policies in *investigationPolicies.xml* and *executive.xml* are defined using a hierarchy of tags. The main policy tag is *policy* which encloses *condition* and *action* tags. *Condtion* tag consists of two *operand* and one *operator* tags. *Action* tag encloses *method* and *parameter* tags which are a method and its parameter that are executed when the condition is true. Some of the policies defined in *investigationPolicies.xml* and *executive.xml* are as follows.

InvestigationPolicies.XML

```
<?xml version="1.0" encoding="utf-8"?>
<InvestigationPolicies>
  <WA  checkingDomain="AllNeighbours" indicatorVector="WA_MaliciousFactor"
       resultType="Scalar"  assumptionFunction="MAX">
```

```
<Policy>
  <Condition>
    <operands>
      <operand>
        <method>
          TheLeastAttachedNeighbour
        </method>
      </operand>
      <operand>
        TRUE
      </operand>
    </operands>
    <operator>
      EQ
    </operator>
  </Condition>
  <Action>
    <method>
      Add
    </method>
    <parameters>
      <parameter>
        1
      </parameter>
    </parameters>
  </Action>
  <Condition>
    <operands>
      <operand>
        <method>
          TheMostTrafficAbsorbentNeighbour
        </method>
      </operand>
      <operand>
```

```
        TRUE
      </operand>
    </operands>
    <operator>
      EQ
    </operator>
</Condition>
  <Action>
    <method>
      Add
    </method>
    <parameters>
      <parameter>
        2
      </parameter>
    </parameters>
  </Action>
  <Condition>
    <operands>
      <operand>
        <method>
          TheFurthestNeighbour
        </method>
      </operand>
      <operand>
        TRUE
      </operand>
    </operands>
    <operator>
      EQ
    </operator>
  </Condition>
  <Action>
    <method>
```

```
          Add
        </method>
        <parameters>
          <parameter>
            1
          </parameter>
        </parameters>
      </Action>
    </Policy>
  </WA>
</InvestigationPolicies>
```

Policy Set A.1: investigationPolicies.xml

ExecutionPolicies.XML

```
<?xml version="1.0" encoding="utf-8"?>
<ExecutionPolicies executiveClass="Executive">
  <WA>
    <Policy>
      <Conditions>
        <condition>
        <operands>
          <operand>
            opinionValue
          </operand>
          <operand>
            4
          </operand>
        </operands>
        <operator>
          EQ
        </operator>
```

```
    </condition>
    <condition>
      <operands>
        <operand>
          opinionValue
        </operand>
        <operand>
          2
        </operand>
      </operands>
      <operator>
        EQGT
      </operator>
    </condition>
    <operator>
      AND
    </operator>
  </Conditions>
  <Action>
    <method>
      SuspendRouting
    </method>
        <parameters>
          <parameter>
            opinions->node
          </parameter>
            5
        </parameters>
  </Action>
  </Policy>
<Policy>
  <Condition>
    <operands>
      <operand>
```

```
            opinionValue
          </operand>
          <operand>
            2
          </operand>
        </operands>
        <operator>
          LEQ
        </operator>
      </Condition>
      <Action>
        <method>
          AddToMalicious
        </method>
        <parameters>
          <parameter>
            opinions->node
          </parameter>
        </parameters>
      </Action>
    </Policy>
  </WA>
</ExecutionPolicies>
```

Policy Set A.2: ExecutivePolicies.xml

Bibliography

[1] Dakshi Agrawal, Kang-Won Lee, and Jorge Lobo. Policy-based management of networked computing systems. *Communications Magazine, IEEE*, 43(10):69–75, 2005.

[2] Kemal Akkaya and Mohamed Younis. A survey on routing protocols for wireless sensor networks. *Ad hoc networks*, 3(3):325–349, 2005.

[3] Farooq Anjum and Petros Mouchtaris. *Security for wireless ad hoc networks*. Wiley. com, 2007.

[4] Marianne Azer, Sherif El-Kassas, and Magdy El-Soudani. A full image of the wormhole attacks-towards introducing complex wormhole attacks in wireless ad hoc networks. *arXiv preprint arXiv:0906.1245*, 2009.

[5] Marianne A Azer, SM El-Kassas, and Magdy S El-Soudani. An innovative approach for the wormhole attack detection and prevention in wireless ad hoc networks. In *Networking, Sensing and Control (ICNSC), 2010 International Conference on*, pages 366–371. IEEE, 2010.

[6] Venkat Balakrishnan, Vijay Varadharajan, and Uday Tupakula. Trust management in mobile ad hoc networks. In *Guide to Wireless Ad Hoc Networks*, pages 473–502. Springer, 2009.

[7] Giuseppe Bianchi. Performance analysis of the ieee 802.11 distributed coordination function. *Selected Areas in Communications, IEEE Journal on*, 18(3):535–547, 2000.

[8] Matt Bishop. Computer security: Art and science. 2003. *Westford, MA: Addison Wesley Professional*, pages 4–12, 2003.

[9] Tracy Camp, Jeff Boleng, and Vanessa Davies. A survey of mobility models for ad hoc network research. *Wireless communications and mobile computing*, 2(5):483–502, 2002.

[10] Srdjan Čapkun, Levente Buttyán, and Jean-Pierre 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*, pages 21–32. ACM, 2003.

[11] M. Carvalho. Security in mobile ad hoc networks. *Security Privacy, IEEE*, 6(2):72–75, 2008.

[12] Hon Sun Chiu and King-Shan Lui. Delphi: wormhole detection mechanism for ad hoc wireless networks. In *Wireless Pervasive Computing, 2006 1st International Symposium on*, pages 6–pp. IEEE, 2006.

[13] Jin-Hee Cho, Ananthram Swami, and Ray Chen. A survey on trust management for mobile ad hoc networks. *Communications Surveys & Tutorials, IEEE*, 13(4):562–583, 2011.

[14] Sun Choi, Doo-young Kim, Do-hyeon Lee, and Jae-il Jung. Wap: Wormhole attack prevention algorithm in mobile ad hoc networks. In *Sensor Networks, Ubiquitous and Trustworthy Computing, 2008. SUTC'08. IEEE International Conference on*, pages 343–348. IEEE, 2008.

[15] Thomas Clausen, Philippe Jacquet, Cédric Adjih, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, Laurent Viennot, et al. Optimized link state routing protocol (olsr). 2003.

[16] Hongmei Deng, Wei Li, and Dharma P Agrawal. Routing security in wireless ad hoc networks. *Communications Magazine, IEEE*, 40(10):70–75, 2002.

[17] DMTF. Dmtf home page, 2013. [Online; accessed 07-July-2013].

[18] Philip England, Qi Shi, Bob Askwith, and Faycal Bouhafs. A survey on trust management in mobile ad-hoc networks. In *Proceedings of the 13th annual post graduate symposium on the convergence of telecommunications, networking, and broadcasting, PGNET*, 2012.

[19] Liljana Gavrilovska and Ramjee Prasad. *Ad hoc networking towards seamless communications*. Springer, 2006.

[20] Ramin Hekmat. *Ad-hoc networks: fundamental properties and network topologies*. Springer, 2006.

[21] Lingxuan Hu and David Evans. Using directional antennas to prevent wormhole attacks. In *NDSS*, 2004.

[22] Y-C Hu, Adrian Perrig, and David B Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1976–1986. IEEE, 2003.

[23] IETF. Ietf home page, 2013. [Online; accessed 07-July-2013].

[24] Pradip M Jawandhiya, Mangesh M Ghonge, MS Ali, and JS Deshpande. A survey of mobile ad hoc network attacks. *International Journal of Engineering Science and Technology*, 2(9):4063–4071, 2010.

[25] Bounpadith Kannhavong, Hidehisa Nakayama, Yoshiaki Nemoto, Nei Kato, and Abbas Jamalipour. A survey of routing attacks in mobile ad hoc networks. *Wireless communications, IEEE*, 14(5):85–91, 2007.

[26] Jeffrey O Kephart and David M Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[27] Issa Khalil, Saurabh Bagchi, and Ness B Shroff. Liteworp: a lightweight countermeasure for the wormhole attack in multihop wireless networks. In *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*, pages 612–621. IEEE, 2005.

[28] James F Kurose and Keith W Ross. *Computer networking*, volume 2. Addison Wesley, 2001.

[29] CK-L Lee, Xiao-Hui Lin, and Yu-Kwong Kwok. A multipath ad hoc routing approach to combat wireless link insecurity. In *Communications, 2003. ICC'03. IEEE International Conference on*, volume 1, pages 448–452. IEEE, 2003.

[30] Wenjia Li and Anupam Joshi. Security issues in mobile ad hoc networks-a survey. *Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County*, 2007.

[31] Janne Lindqvist. Counting to infinity. In *Seminar on Internetworking, Helsinki University of Technology Telecommunications, Software and Multimedia Laboratory*, 2004.

[32] Songbai Lu, Longxuan Li, Kwok-Yan Lam, and Lingyan Jia. Saodv: a manet routing protocol that can withstand black hole attack. In *Computational Intelligence and Security, 2009. CIS'09. International Conference on*, volume 2, pages 421–425. IEEE, 2009.

[33] Zheng Lu and Hongji Yang. *Unlocking the power of OPNET modeler*. Cambridge University Press, 2012.

[34] Emil C Lupu and Morris Sloman. Conflicts in policy-based distributed systems management. *Software Engineering, IEEE Transactions on*, 25(6):852–869, 1999.

[35] Sergio Marti, Thomas J Giuli, Kevin Lai, Mary Baker, et al. Mitigating routing misbehavior in mobile ad hoc networks. In *International Conference on Mobile Computing and Networking: Proceedings of the 6 th annual international conference on Mobile computing and networking*, volume 6, pages 255–265, 2000.

[36] Matt Murphy. Top mobile internet treands @ONLINE, November 2011.

[37] Charles E Perkins and Elizabeth M Royer. Ad-hoc on-demand distance vector routing. In *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA'99. Second IEEE Workshop on*, pages 90–100. IEEE, 1999.

[38] W Timothy Polk and Lawrence E Bassham. Guide to the selection of anti-virus tools and techniques. Technical report, DTIC Document, 1992.

[39] Mohammad Sadeghi and Saadiah Yahya. Analysis of wormhole attack on manets using different manet routing protocols. In *Ubiquitous and Future Networks (ICUFN), 2012 Fourth International Conference on*, pages 301–305. IEEE, 2012.

[40] T Sakthivel and RM Chandrasekaran. Detection and prevention of wormhole attacks in manets using path tracing approach. *European Journal of Scientific Research, ISSN*, pages 240–252, 2012.

[41] Subir Kumar Sarkar, TG Basavaraju, and C Puttamadappa. *Ad hoc mobile wireless networks: principles, protocols and applications*. CRC Press, 2007.

[42] Morris Sloman. Policy driven management for distributed systems. *Journal of network and Systems Management*, 2(4):333–360, 1994.

[43] Ning Song, Lijun Qian, and Xiangfang Li. Wormhole attacks detection in wireless ad hoc networks: A statistical analysis approach. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 8–pp. IEEE, 2005.

[44] Heng-jun WANG, Ya-di WANG, and Qi ZHANG. Survey of trust management in mobile ad hoc networks [j]. *Journal of Computer Applications*, 5:035, 2009.

[45] Wikipedia. Bellmanford algorithm — wikipedia, the free encyclopedia, 2013. [Online; accessed 9-July-2013].

[46] Wikipedia. Dynamic source routing — wikipedia, the free encyclopedia, 2013. [Online; accessed 18-July-2013].

[47] Wikipedia. Personal area network — wikipedia, the free encyclopedia, 2013. [Online; accessed 18-June-2013].

[48] Wikipedia. Vaccine — wikipedia, the free encyclopedia, 2013. [Online; accessed 25-June-2013].

[49] Bing Wu, Jianmin Chen, Jie Wu, and Mihaela Cardei. A survey of attacks and countermeasures in mobile ad hoc networks. In *Wireless Network Security*, pages 103–135. Springer, 2007.

[50] Yanchao Zhang, Wenjing Lou, Wei Liu, and Yuguang Fang. A secure incentive protocol for mobile ad hoc networks. *Wireless Networks*, 13(5):569–582, 2007.

Curriculum Vitae

Name:	Arash Tajalli-Yazdi
Post-Secondary Education and Degrees:	B.Sc. in Computer Software Engineering Islamic Azad University of Mashhad (IAUM) 2001-2006
	M.Sc. in Computer Science Western University London, ON 2011 - 2013
Related Work Experience:	Teaching Assistant Computer Science Department, Western University London, Ontario, N6A5B7, Canada Western University 2011 - 2012
	Research Assistant Computer Science Department, Western University London, Ontario, N6A5B7, Canada Western University 2011 - 2013