Analysis of Black Hole Attack in Wireless Ad-hoc Networks

Seyed Abouzar Hosseini

am Lehrstuhl für Telematik

Masterarbeiten des Zentrums für Informatik an der Georg-August-Universität Göttingen
February 2014
Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Göttingen, den 28 February 2014
Master’s Thesis

Analysis of Black Hole Attack in Wireless Ad-hoc Networks

Seyed Abouzar Hosseini

February 2014

Supervised by Prof. Dr. Dieter Hogrefe

(Dr. Parisa Memarmoshrefi)

Telematics Research
Institute for Computer Science
Georg-August-University of Göttingen, Germany
Abstract

Wireless ad hoc networks have been a hot research topic in recent years. They are relatively a new type of network that do not require any infrastructures. Therefore, they can operate without any need for a central administration in a decentralized manner. This lack of requirement for infrastructure makes them flexible and popular because then can provide connectivity in situations where previously was not possible. This lack of administration also comes with disadvantage of extra vulnerability in addition to vulnerabilities that are inherited from wired and wireless networks. Thus, the security is a major concern for wireless ad hoc networks.

Among the attacks that are specific to wireless ad hoc networks, Black hole attack is a famous one. In this attack, attackers try to attract data packets in the network and drop them, which lead to the Denial of Service (DoS) attack. In this work first the definition, mechanism and different implementation of black hole attack is discussed and explained. Later the impact of this attack, on different network topologies in various scenarios, is analyzed through series of simulations in MATLAB.

Keywords: Wireless ad-hoc network, black hole attack, simulation
Table of Contents

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Related works

2 Ad hoc Wireless Networks

2.1 Applications

2.2 Advantage and Disadvantages

2.2.1 Advantages of ad hoc networks

2.2.2 Disadvantages of ad hoc networks

3 Routing

3.1 Update mechanism

3.1.1 Proactive or table-driven routing protocols

3.1.2 Reactive or on-demand routing protocols

3.1.3 Hybrid routing protocols

3.2 Temporal information

3.2.1 Based on past temporal information

3.2.2 Based on future temporal information

3.3 Topology information

3.4 Resource utilization

3.4.1 Power-aware routing

3.4.2 Geographical information assisted routing

3.5 DSR, AODV and OLSR
3.6 AODV .........................................................................................................................16
3.7 OLSR..........................................................................................................................17
4 Security in Ad Hoc Wireless Networks ................................................................. 19
  4.1 Vulnerabilities .........................................................................................................19
  4.2 Attacks ....................................................................................................................20
    4.2.1 Network layer attacks ....................................................................................22
5 Black Hole Attack ......................................................................................................26
  5.1 Single and Collaborative (Co-operative) Black Hole Attacks .........................27
  5.2 Black Hole Attack in Different Routing Protocols (Proactive and Reactive) 27
    5.2.1 Black Hole Attack in Reactive (on-demand) Routing Protocols ..................27
      5.2.1.1 Internal and External Black Hole Attacks ..............................................27
      5.2.1.2 RREQ and RREP Black Hole Attacks ................................................28
    5.2.2 Black Hole Attack in Proactive (Table-driven) Routing Protocols ..........30
6 Implementation of Simulator .....................................................................................32
  6.1 Simulation Tool ......................................................................................................32
    6.1.1 MATLAB .....................................................................................................32
  6.2 Modeling of Network ............................................................................................33
  6.3 Base Code .............................................................................................................33
  6.4 Added Code ........................................................................................................34
7 Simulations and Results ..........................................................................................37
  7.1 Performance Metrics ...........................................................................................37
  7.2 Evaluation of The Simulation ..............................................................................37
  7.3 Simulations Setups ...............................................................................................38
  7.4 Results and Discussion ......................................................................................39
8 Conclusion and Future Work ..................................................................................45
  8.1 Future work ..........................................................................................................45
References ..................................................................................................................47
Appendices ..................................................................................................................51
Source Codes..............................................................................................................51
List of Figures

Figure 3.1 – Ad hoc routing protocols classification ..........................10
Figure 3.2 – Dynamic Source Routing Classification ......................... 14
Figure 3.3 – Propagation of Route Request packet and Route Reply packets in AODV . . 17
Figure 3.4 – Multipoint Relay (MPR) Selection in OLSR ......................18
Figure 4.1 – Classification of ad hoc attacks ..........................21
Figure 5.1 – Black hole attack .................................................26
Figure 5.2 – Black hole attack formed by fake RREP .........................29
Figure 5.3 – Black hole attack formed by fake RREQ .......................30
Figure 6.1 – Implemented web GUI for parameters input ......................34
Figure 7.1 – Tested topologies for evaluation of simulator ....................38
Figure 7.2 – Grid topology that is used for simulation .......................39
Figure 7.3 – Packet delivery ratio of different scenarios .....................40
Figure 7.4 – Average Packet Delivery Ration of all simulated scenarios ....40
Figure 7.5 – Throughput of different scenarios ..........................41
Figure 7.6 – Average Throughput of all simulated scenarios ...............42
Figure 7.7 – Goodput of different scenarios ....................................42
Figure 7.8 – Average Goodput of all simulated scenarios ................... 43
Figure 7.9 – Average end-to-end delay of all simulated scenarios ..........44
List of Tables

Table 2.1 - Mobile ad hoc network applications ........................................... 5

Table 3.1 – On-demand routing protocols vs. Table-driven routing protocols .... 12

Table 4.1 - Attacks on ad hoc wireless networks based on their OSI layers ....... 23
1 Introduction

Communication has always been important for humans, and through the history people has tried to simplify and advance the ways that they can communicate and exchange information with each others. Internet is now everywhere and wired and wireless networks connect people and computers all around the world. However, this connectivity comes at cost of having proper infrastructure and administration for the network in place. This is why there is either no connectivity or very poor connectivity in non-urban and even in non-commercial areas. This is where ad hoc networks come into attention.

Ad hoc networks are relatively new type of networks that can bring connectivity into new places and situations where it was not possible with traditional networks. For example, with ad hoc it is possible to communicate after an earthquake where many of the infrastructures are destroyed, or it is possible to get notified immediately when a fire starts in the middle of a jungle. Also, ad hoc networks are one of the key technologies that help the idea of Internet of Things become a reality. Internet of Things tries to make devices smarter by connecting them together and let them exchange information. Smart homes and smart roads are two famous examples for the usages of the Internet of Things. In a smart home all electronic devices can communicate with each other and even services outside a house in order to optimize their usages and bring more quality to the life. Smart road is the idea that all the vehicles can be aware of their surrounding and be able to communicate with other vehicles on the road. These will help to make smart vehicles in order to reduce accidents and traffics.

Ad hoc networks are decentralized networks that do not require pre-existing network infrastructure such as routers and access points; therefore, each node in the network can act autonomously. These characteristics give admiring flexibility to this type of networks, and that is why ad hoc networks have become a hot research topic in recent years.

Like any other technology ad hoc network comes with set of advantages and disadvantage. Although they are capable of providing many new benefits, they still face different challenges. Among all the issues such as limited resources and routing optimization, the security is most important challenge in ad hoc networks. Lack of central management and monitoring make detection and prevention of various attacks very difficult. Many challenges have to be addressed in order to insure confidentiality, integrity, authentication, availability and authorization in these networks. Because an ad hoc network uses the technologies that are used in regular wireless and mobile networks, it inherits their security threats and vulnerabilities. In addition, having its own characteristics introduces new type of vulnerabilities to these networks.
One of the attacks that is specific to the ad hoc networks is Black hole attack. This attack takes advantage of decentralized characteristic of the network and cause packet loss in the network. As the result, this attack can reduce the network throughput, and at worst case makes a part of the network inaccessible. This thesis has its main focus on this attack. Therefore, it will be discussed in details how this attack works, and what are the impacts of it on a network.

This thesis is organized in following chapters. Chapter 2 introduces ad hoc network in details by defining and discussing characteristics and boundaries of ad hoc network. Then in Chapter 3 routing protocols are discussed since they are key differentiator of the wireless ad hoc networks. Chapter 4 discusses major security threats in ad hoc networks, and Chapter 5 investigates the black hole attack in detail. Chapter 6 introduces the implemented code that is used for simulations in this work, and Chapter 7 discusses the simulations and the results. Finlay, the conclusion of the work and the possibilities for future works is presented in Chapter 8.

1.1 Related works

Most of the works have compared the effect of black hole attack on different routing protocols [20, 17, 19]. Joy et al. [20] studied changes of network load and throughput in Optimized Link State Routing (OLSR) and Ad hoc On-Demand Distance Vector Routing (AODV) under black hole attack. Kumar et al, [20] used end-to-end delay, packet delivery ratio and network overhead metrics in order to compare performance of AODV and IAODV (Improved AODV) in presence of the attack. Kaur et al. [19] compared impact of the attack on OLSR, AODV and Zone Routing Protocol (ZRP). They used packet delivery ratio, jitter and network throughput metrics. Others instead of comparing different routing protocols only focused on some aspect of a specific routing protocol. Salehi et al. [28] studied only OLSR and they also proposed a new black hole implementation through modification of Topology Control messages in OLSR, which is called TC-Black-Hole attack.

There are studies on networks with mobility and their behaviors in presence of black hole attack [30, 18]. They only examined the performance of the network by looking into packet delivery ratio and throughput of the network.

Malicious nodes collusion in black hole attack and countermeasures against that has been studied in [15, 16, 25].

There are various surveys that summarized the behavior of different routing protocols in black hole attacks [36, 34, 6, 13], and some surveyed the prevention
ways of black hole [35]. In a broader topic, [31] and [14] surveyed all attacks in mobile Ad hoc Network (MANET).

The main focus of this work is on the proportion of the network that can be controlled by attackers. The simulations are implemented in MATLAB. The routing protocol is DSR. Also, packet delivery ratio, end-to-end delay, throughput and goodput are used metrics for evaluation of network performance in presence of different percentages of malicious nodes.
2 Ad hoc Wireless Networks

The best way to define wireless ad hoc network is to look at its definition, and find out what ad hoc means. According to Oxford dictionary ad hoc means “created or done for a particular purpose as necessary”. In the other words, it is not meant to be permanent and general. Wireless ad hoc network is a type of wireless network, which is built on top of wireless standards, and it is used for scenarios where devices connect to each other based on their requirements. In this way, there is no need for a central device to administrate the communication; therefore, devices can talk to each other directly whenever they need to. Mobile Ad-hoc Networks (MANETs) and Wireless Sensor Networks (WSNs) are two specific kind of the wireless ad-hoc network.

2.1 Applications

The infrastructure-less nature of ad hoc networks opens a vast range of application to this kind of networks. Basically wherever building an infrastructure is impossible, costly or difficult the ad hoc network is a well replacement for traditional infrastructure based networks.

The following describes some of the most notable applications of the mobile ad hoc networks, and table 2.1 provides a summary of these applications.

Military: Like it or not the very first use of ad hoc networks were in military [33]. The most important usage could be in battlefields were there are almost everything is dynamic. There are no time or possibility to setup up any infrastructure; therefore, the best way to establish communications between people is mobile ad hoc networks. Also, by recent rapid improvement of automated system in the battle field they can also take benefit of ad hoc network and communicate between each other and their command center through mobile ad hoc networks.

Rescue and emergency: There are two main scenarios. First, when a disaster such as earth quick, fire or hurricane happens in remote areas where there are no access to normal infrastructures for communications, ad hoc network are the most suitable choice in these situations. They can be setup easily and quickly without any infrastructure requirements.

Second, when a disaster happens in populated area and the usual infrastructures are become inaccessible, either by getting damaged physically or overloaded with many requests from different networks. In both cases they are not fully functional and useful; however, in this situations communications between
rescue teams and people in the area are quite critical. Therefore, again the best way of communication is mobile ad hoc network. Whenever the mission is over or situation became under control and traditional infrastructure resumes their normal activities, the ad hoc networks can be removed easily.

**Personal Area Network (PAN):** These are networks with small range (typically smaller than 100m) that do not require any fix infrastructures. It is mainly used for connecting personal devices such as smartphone, laptop and tablet to each other in order to exchange personal data. Different technologies can be used in order to create a PAN, but based on definition Bluetooth is the most famous example of them.

**Wireless Sensor Networks (WSN):** It is a network of distributed autonomous sensors that is used for monitoring. Each node is equipped with at least a type of sensor that gathers information from its surroundings. Usually nodes only gather information and they use the network to transfer their data to a central location for further actions. Because these networks usually try to cover large areas, they can consist of thousands of nodes. These nodes are usually very cheap and therefore have very limited resources. They are very good for monitoring in industries and manufactures, environmental monitoring, health care monitoring and disaster prevention (or at least fast response).

**Wireless Mesh Networks (WMN):** It is a wireless network that is organized in a mesh topology. Wireless mesh networks are defined in IEEE 802.11s, which is an amendment of IEEE 802.11. The applications of WMNs are like Local Area Networks (LANs) and Metropolitan Area Networks (MANs).

<table>
<thead>
<tr>
<th>Application</th>
<th>Possible scenarios/services</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tactical networks</strong></td>
<td>• Military communication and operations</td>
</tr>
<tr>
<td></td>
<td>• Automated battlefields</td>
</tr>
<tr>
<td><strong>Emergency services</strong></td>
<td>• Search and rescue operations</td>
</tr>
<tr>
<td></td>
<td>• Disaster recovery</td>
</tr>
<tr>
<td></td>
<td>• Replacement of fixed infrastructure in case of environmental disaster</td>
</tr>
<tr>
<td></td>
<td>• Policing and fire fighting</td>
</tr>
<tr>
<td></td>
<td>• Supporting doctors and nurses in hospitals</td>
</tr>
<tr>
<td><strong>Commercial and civilian environments</strong></td>
<td>• E-commerce: electronic payments anytime and anywhere</td>
</tr>
<tr>
<td></td>
<td>• Business: dynamic database access, mobile offices</td>
</tr>
<tr>
<td></td>
<td>• Vehicular services: road or accident guidance, transmission of road and weather conditions, taxi</td>
</tr>
</tbody>
</table>

Table 2.1 - Mobile ad hoc network applications [7]
| Home and enterprise networking | • Home/office wireless networking  
|                               | • Conferences, meeting rooms  
|                               | • Personal area networks (PAN), personal networks (PN)  
|                               | • Networks at construction sites  
| Education                     | • Universities and campus settings  
|                               | • Virtual classroom  
|                               | • Ad hoc communications during meeting or lectures  
| Entertainment                 | • Multi-use games  
|                               | • Wireless P2P networking  
|                               | • Outdoor internet access  
|                               | • Robotic pets  
|                               | • Theme parks  
| Sensor networks               | • Home applications: smart sensors and actuators embedded in consumer electronics  
|                               | • Body area networks (BAN)  
|                               | • Data tracking of environmental conditions, animal movements, chemical/biological detection  
| Context aware services        | • Follow-on services: call-forwarding, mobile workspace  
|                               | • Information services: location specific services, time dependent services  
|                               | • Infotainment: touristic information  
| Coverage extension            | • Extending cellular network access  
|                               | • Linking up with the internet, intranets, etc.  

### 2.2 Advantage and Disadvantages

Main wireless ad hoc characteristics can be listed as following [7]:

- Autonomous and infrastructure
- Multi-hop routing
- Dynamic network topology
- Device heterogeneity
- Energy constrained operation
- Bandwidth constrained variable capacity links
- Limited physical security
- Network scalability
• Self-creation, self-organization and self-administration

Like any other technology each of these characteristics has its advantages and disadvantages. For example, while lack of infrastructure brings many advantages such as autonomy and flexibility to the network, in the same time it adds more vulnerability and security challenges to the network. These advantages and disadvantages are discussed in the following.

2.2.1 Advantages of ad hoc networks

The reason that wireless ad hoc network has been a very hot research topic in recent years is its advantages. They together make it possible to use this type of network in situations where previously was not possible. Some major advantages are introduced as below:

**Infrastructure-less:** Independency from infrastructures can be seen as the main advantage of the ad hoc network because they provide wide range of advantages such as less cost, easiness of setup, accessibility to the network. Also, having an infrastructure-less network can bring connectivity and communication to areas and situations where installing infrastructures are either impossible or very difficult.

**Lower cost:** Traditionally a big portion of expenses has been related to building, purchasing, installing and maintaining infrastructures for a network. Ad hoc networks do not require any infrastructure; therefore, they have much less cost than other type of networks.

**Decentralized:** Lack of central administration comes with its own advantages and disadvantages. On the positive side, nodes in ad hoc networks are not dependable only on one node and they can act and communicate without it. This is good when a base station is compromised, malfunction or just become inaccessible. Unlike ad hoc networks, in other networks any changes to a base station can affect all the nodes that are connecting to that node, and they can all get compromised or lose their connectivity to the rest of the network.

**Easy and rapid setup:** When no infrastructure is required, no planning and setup is also needed for the network. Nodes in ad hoc networks are autonomous and they not need sophisticate and complex central administration setup.

**Mobility:** The benefit of mobility is obvious when considering the mentioned application of the ad hoc networks in military, emergency rescues and etc. Two nodes can communicate with each other as long as they are in each other ‘s radio cover or if they can found common middle nodes between themselves. However, not necessarily all type of ad hoc networks can enjoy the benefit of mobility, and it is restricted to MANET.
2.2.2 Disadvantages of ad hoc networks

While the lack of infrastructure and central authority provides a wide range of flexibilities to the network, these characteristics pose some limitations and vulnerabilities to the network. In addition, ad hoc wireless networks inherit many disadvantages off wireless networks such as lower reliability than wired media, interferences and limited physical security.

**Higher error rate:** Wireless waves are transmitted into open air unlike wired networks; therefore they are vulnerable for being disrupted. Depend on the environment shadowing, reflection, scattering, fading, refraction or diffraction of wireless waves are possible. These or combination of these can disrupt the wave and cause packet error.

**Lower data rate:** Again characteristics of wireless wave prevent wireless communication to transmit data better than wired communication. Although, a higher frequency can transmit more data, it is more vulnerable to interference and it does not perform well in long range [5].

**Latency:** Because there are no central hubs like switches and routers in the ad hoc networks, routing has to be handle in a distributed manner through the other nodes. Considering rapid changes in the topology of the network and percentages of link failures routing could suffer from serious delays. This is even worse when the network is big and has too many nodes.

**Scalability:** Ad hoc networks are not as scalable as regular wireless networks because ad hoc networks do not allow the same kinds of aggregation techniques that are available to standard Internet routing protocols [5].

**Security:** Lack of central administration, limited resources, dynamic topology and lack of infrastructure that could provide a defense line or monitoring pose many security issues and challenges to ad hoc networks. In addition, ad hoc networks suffer from wireless limited physical security (in comparison to wired networks). Therefore, the ad hoc networks are susceptible to attacks ranging from passive attacks such as eavesdropping to active attack such as interfering. Ad hoc security issues are discussed in detail in Chapter 4.
3 Routing

The routing protocol is the heart of the ad hoc wireless networks. From software point of view it is the key part that differentiate ad hoc wireless networks from the regular infrastructure-based wireless networks.

Routing protocols simply provide a way for a source node to find a path to a destination node for transmission of packet data on the path. However, traditional wired routing protocols cannot directly be used in ad hoc network because of wireless nature of ad hoc networks and their dynamic topology. Also, regular wireless routing protocols cannot directly be applied on ad hoc network due to the lack of infrastructures and central administration. Thus, new routing protocols has been designed and proposed for ad hoc networks. Before discussing these protocols, a quick look at general issues and characteristics of a routing protocol in ad hoc network would be helpful.

Regardless of a network type a routing protocols has to have certain characteristics such as being loop-free and having fast convergence; also, in wireless networks there are additional issues such as shared medium and hidden and exposed terminal problems. In addition to inheriting all those characteristics and issues, ad hoc routing protocols has to deal three major characteristics of the ad hoc networks: mobility, self-administration and limited resources. Mobility causes rapid changes in topology of the network; therefore, routing protocol has to be aware of all the changes and failure in the network. Lack of central administration and infrastructure in one hand and limited resources such as battery, computing power and etc. in the other hand pose more challenges to the routing protocol for being efficient and optimized.

Therefore, a routing protocol in ad hoc wireless networks at least has to have the following characteristics:

- Scalable
- Fully distributed
- Adaptive to dynamic topology
- Minimum overhead
- Optimally use each node’s resources:
  - Battery
  - Memory
  - Computing power
  - Bandwidth
- Loop-free
Of course the above-mentioned list does not cover all the aspect and characteristics of an ad hoc routing protocol, but they are the very fundamental and must have characteristics.

The routing protocols in ad hoc wireless networks can be categorized based on the followings [8]:

1. Update mechanism (routing table)
2. Temporal information
3. Topology information
4. Resource utilization

Routing protocols in each of these categories share some basic attributes and characteristics; however, some of them may fall into more than one category due to their hybrid nature. Each of these categories is briefly reviewed in the followings, and their hierarchy is shown in the figure 3.1.

![Figure 3.1 – Ad hoc routing protocols classification [8]](image)

### 3.1 Update mechanism

#### 3.1.1 Proactive or table-driven routing protocols

These routing protocols are extensions of wired routing protocols [23]. In this type of protocol every node maintain routing tables, which they contain information regarding the network topology. Through periodic updates these routing tables are kept up-to-date and fresh. Whenever a node requires a path to
another node, the best path is calculated and retrieved locally from the node's routing tables. All routes in the network are always available to all nodes which leads to low latency in route finding and less delay in route setup process in this type of routing protocols [8]. However, this speed increase comes at cost of more overhead control and additional resource usage because the dynamic nature of ad hoc network requires too many updates in order to keep the routing tables fresh.

The followings are some of the table-driven routing protocols:

- Destination Sequence Distance Vector (DSDV)
- Wireless Routing Protocol (WRP)
- Source-Tree Adaptive Routing (STAR)
- Optimized Link State Routing (OLSR)

### 3.1.2 Reactive or on-demand routing protocols

Routing protocols under this category do not maintain the topology information of the network through a periodic update. As they name suggests they work reactively and whenever there is a demand they request a route from the network. In oppose to table-driven protocols these protocols try to avoid unnecessary periodical information exchange in order to optimize the resources usages. This comes at cost of having a high latency time in route finding, since the route discovery starts with the request.

The followings are some of the on-demand routing protocols:

- Ad hoc On-demand Distance Vector (AODV)
- Dynamic Source Routing (DSR)

DSR is discussed in detail in the section 3.5.1. Also, the simulations on this work are based on DSR.

### 3.1.3 Hybrid routing protocols

Proactive and reactive routing protocols each have their own advantages and disadvantages, which a comparison of the two can be found in the table 3.1. Therefore, some of the routing protocols try to combine advantages from the both proactive and reactive in order to eliminates their disadvantages, which are called hybrid routing protocols.

A certain distance from the node, or certain geographical regions are called routing zone of a node. Any node in the routing zone of a node treated with proactive (table-driven) approach. They exchange they routing information periodically, and also update each other in case of any changes. The rest of the
nodes in the network will be accessed with reactive (on-demand) approach. When destination is not in the routing zone of a node, the node start to send Route Request packets in order to obtain a path to the destination.

- Core-Extraction Distributed Ad hoc Routing (CEDAR) [8]
- Zone Routing Protocol (ZRP)
- Zone-based Hierarchical Link State (ZHLS) [3]

**Table 1.1 – On-demand routing protocols vs. Table-driven routing protocols** [26]

<table>
<thead>
<tr>
<th>Parameters</th>
<th>On-demand</th>
<th>Table-driven</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Availability of routing information</strong></td>
<td>Available when needed</td>
<td>Always available regardless of need</td>
</tr>
<tr>
<td><strong>Routing philosophy</strong></td>
<td>Flat</td>
<td>Mostly flat, except for Cluster-head Gateway Switch Routing (CGSR)</td>
</tr>
<tr>
<td><strong>Periodic route updates</strong></td>
<td>Not required</td>
<td>Required</td>
</tr>
<tr>
<td><strong>Coping with mobility</strong></td>
<td>Use localized route discovery as in Associativity-Based Routing (ABR) and Signal Stability Routing (SSR)</td>
<td>Inform other nodes to achieve a consistent routing table</td>
</tr>
<tr>
<td><strong>Signaling traffic generated</strong></td>
<td>Grows with increasing mobility of active routes (as in ABR)</td>
<td>Greater than that of on-demand routing</td>
</tr>
<tr>
<td><strong>Quality of service support</strong></td>
<td>Few can support QoS, although most support shortest path</td>
<td>Mainly shortest path as the QoS metric</td>
</tr>
</tbody>
</table>

### 3.2 Temporal information

Ad hoc wireless networks have dynamic topology and path change happens more frequently than the other type of the networks. This type of routing protocols find a route based on the temporal information that they gather from the network.
3.2.1 Based on past temporal information

This type of protocols use current or the past link status between nodes. They take into the account the lifetime of the wireless links and the lifetime of the paths in order to calculate the shortest path in the current time.

3.2.2 Based on future temporal information

These protocols try to foresee the future information about the status of the links. They use different factors such as remaining lifetime of a node (based on the available battery and their discharging rate), prediction of location and prediction of the link’s availability [23].

3.3 Topology information

A 32-bit IP address is known as a structured or hierarchical address as opposed to a flat or nonhierarchical address [21]. Traditionally all the routing protocols uses a hierarchal routing information. The main reason that hierarchical scheme is chosen is its ability to handle large number of addresses (up to 2 to power of 32 or 4,294,967,296). This is a big advantage for routers because no additional state information is required to be stored on them. However, in an ad hoc network, number of nodes is relatively small and there is even no infrastructure (router). Therefore, both flat and hierarchical addressing can be used in the ad hoc routing protocols. Based on the addressing scheme and topology that a routing protocol is defined on, they can be categorized into one of the followings:

- Flat topology routing protocols
- Hierarchical topology routing protocols:

In [21] a discussion of the two addressing schemes is presented. While flat addressing may be less complicated and easier to use, there are doubts as to its scalability.

3.4 Resource utilization

There are different network resources that can assist routing protocols. The followings highlight two major resources in this classification.

3.4.1 Power-aware routing

These protocols by calculating the available power and the cost of the power for each routing try to optimize the power consumption. Therefore, they make their
routing decision in order to minimize the power consumption either locally or globally [23].

3.4.2 Geographical information assisted routing

These routing protocols try to reduce the control overhead on the network by utilizing the geographical information of each node in their routing algorithm.

3.5 DSR, AODV and OLSR

In this section some of the most well-known routing protocols are discussed. As it will be discussed in the chapter 5, black hole is implemented differently in each of these protocols. Also, in most researches mainly these protocols are used for evaluation of the black hole attack.

3.5.1 DSR

![Diagram](image)

**Figure 3.2 - Dynamic Source Routing Classification**

The *Dynamic Source Routing* protocol (DSR) [15] is a simple and efficient on-demand routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes. It was developed as a part of the Monarch Project at Carnegie Mellon University. Their goal was to create a routing protocol for wireless ad hoc networks with very low overhead yet with rapid response to the changes in the network. DSR is designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes; thus, it allows network to be completely self-organizing and self-configuring.
The DSR protocol is composed of two mechanisms that work together to allow the discovery and maintenance of source routes in the ad hoc network:

- **Route Discovery:** When a node (Source) wants to communicate with another node (Destination) and there is no known path exists in the source node’s cache, Route Discovery is initiated. The source node starts Route Discovery by broadcasting a Route Request (RREQ) message. Every other node in the network that receive the RREQ message also broadcast the message until it reaches the destination node or an intermediate node has a fresh route to the destination. Afterward, the node that has valid information about the destination generates a Route Reply (RREP) message and sends it to the source node. When the source node receives the RREP, it updates its cache and the Route Discovery process is over. In addition, a node may learn and cache multiple routes to any destination [16]. This allows rapid response to any route changes, and avoids the overhead of needing to perform a new Route Discovery each time a route in use breaks.

- **Route Maintenance:** When Route Discovery is over and the source node start transferring data to its destination, DSR check the route during the transmission and make sure the route is not broken. If DSR detects any changes in the network topology that breaks the route during data transmission, the source node is informed and a new Route Discovery will be executed. In order to perform the route maintenance, every node that involves in transmitting data from a source node to its destination is responsible to confirm the data delivery to its next hop. If a node does not receive a delivery confirmation to its next hop, it will generate a Route Error (RERR) message and sends it to the source node. After receiving the RERR, source node reinitiates the Route Discovery again in order to find a new route to its destination.

In [2], Bhimla et al run series of simulation on DSR and they claim that DSR is suitable for networks with moderate mobility rate. Also, because it has low overhead, it is suitable for low bandwidth and low power network. The following are some of the advantages and disadvantages of DSR:

**DSR Advantages:**

- A route is established only when it is required
- The intermediate nodes also utilize the route cache information efficiently to reduce the control overhead.

**DSR Disadvantages:**

- The route maintenance mechanism does not locally repair a broken link.
- The connection setup delay is higher than in table-driven protocols.
• The routing overhead is directly proportional to the path length, which could cause a considerable routing overhead.

3.6 AODV

The Ad Hoc On-Demand Distance-Vector (AODV) is reactive routing protocol that incorporates the features of both DSDV and DSR. AODV was proposed by Perkins et al as enhancement to the DSDV and DSR, and later standardized by IETF in collaboration with Nokia Research Center, University of California, Santa Barbara and University of Cincinnati [25]. DSDV is a proactive routing protocol, which is based on Bellman-Ford algorithm. The Bellman-Ford algorithm does not prevent routing loops and suffers from the count-to-infinity problem. While AODV uses DSDV features, by avoiding the Bellman-Ford counting to infinity problem, it offers a loop-free routing with a quick convergence when the ad hoc network topology changes.

Similar to DSR the routing in AODV is accomplished in two phases: route discovery and route maintenance.

• **Route Discovery**: A node initiate route discovery by broadcasting Route Request (RREQ) message when it does not have a valid route to a destination in its cache. Any node that receives the RREQ and does not have any fresh route to the destination broadcasts this RREQ. Either destination node or an intermediate node with a fresh route to destination replies to the RREQ by unicasting a Route Reply (RREP) message to the source node. As the RREP is routed back along the reverse path, intermediate nodes along this path set up forward path entries to the destination in their routing tables. When the RREP reaches source node, a route from source to destination node is established. Route discovery is illustrated in Figure 3.3.

While DSR relies on time to live (TTL) value for loop detection, AODV uses sequence numbers in order to guarantee loop-free routes. AODV assigns each node that requests a route a Sequence Number and it is increased every time the node makes a new request. Higher sequence number is assumed to be more accurate information and whichever node sends the highest sequence number, its information is considered most up to date and route is established over this node by the other nodes.
Route Maintenance: Once the route discovery is over and a route is established, it is time for Route Maintenance. When an active route breaks or a failure is detected by any node during the transmission, Route Error (RERR) message is flooded to the network till it reaches the source node. Then it would be decision of the source node to stop the data transmission or reinitiate another route discovery process.

While AODV is an efficient reactive routing protocol, it has some limitations as well. The following are some of the AODV limitations [20]:

- If the sequence number of source node is lower than that of intermediate nodes, it may lead to inconsistent routes.
- Multiple route reply packets and periodic beaconing may result in heavy routing overhead.
- The overall performance starts degrading as network grows.

3.7 OLSR

The Optimized Link State Routing Protocol (OLSR) is a proactive link-state routing protocol for mobile ad hoc networks. It was proposed by Jacquet et al. as an optimization of the classical link state algorithm, and it is tailored to the requirements of a mobile wireless LAN [12].

In OLSR each node maintains topology information about the network by periodically exchanging link-state messages. Although, the general idea of the OLSR is same as the traditional link state protocol, the OLSR differentiate itself by minimizing the size of control packets and the number of rebroadcasting nodes during each route update. In order to reduce the number of rebroadcasting routing packets in the network, OLSR introduces Multipoint relays (MPRs). For any given node its MPRs are subset of its one-hop neighbors that cover all the second hop neighbors of that node. Only MPRs broadcast the touring packets; thus, any node that is not a MPR just can receive and process the routing packets, but it does not rebroadcast them.
To select the MPRs, each node periodically broadcasts its HELLO messages that contain:

- The list of addresses of the neighbors to which there exists a valid bi-directional link.
- The list of addresses of the neighbors which are heard by this node (a HELLO has been received) but the link is not yet validated as bi-directional (if a node finds its own address in a HELLO message, it considers the link to the sender node as bi-directional).

All one-hop neighbors receive these HELLO messages, but do not broadcast them any further. From the list of nodes in the hello messages, each node selects a subset of one-hop neighbors, which covers all of its two hop neighbors. Then each node determines an optimal route (in terms of hops) to every known destination using its topology information (from the topology table and neighboring Table), and stores this information in a routing Table [6]. Figure 3.4 shows an example MPR selection in OLSR. Node N, in figure 3.4, can select nodes B, D, F, H to be the MPR nodes.

![Multipoint Relay (MPR) Selection in OLSR](image-url)
4 Security in Ad Hoc Wireless Networks

One of the main obstacles in front of wide availability and use of ad hoc wireless network is concerned to their securities. Some of the challenges in their routing protocols are discussed in the previous chapter, and in this chapter all of their vulnerabilities plus the major attacks on them will be reviewed.

4.1 Vulnerabilities

Ad hoc wireless networks are more vulnerable to attacks than wired network because of their wireless nature. Also, because they do not have a central administration, they are even more vulnerable than other types of wireless network. Some of the most important challenges that makes ad hoc wireless network vulnerable are discussed in the following.

**Open medium:** wireless links and communication among the network’s nodes are open to every one in their range, which this makes the network susceptible to attacks such as eavesdropping and active interference. This open nature makes it easier for attackers to attack the network in comparison to the wired network, which attacker need to get physical access to the network. Also, wireless networks usually have lower bandwidth than wired networks; therefore, attackers could just consume the bandwidth and cause disruption in the network [29].

**Dynamic topology:** mobility cause rapid changes in the network. As a result of mobility nodes become available and unavailable very frequently. Aside from the problems that it could cause for the routing protocol, this makes detection of malicious activities much harder. Anomalies cannot be easily differentiated from normal activities in the networks. Malicious nodes could easily take advantage of this feature and reintroduce themselves into the network after a while in order to avoid detection and penalties.

**Cooperativeness:** By default most of the routing protocols in ad hoc networks assume that all the nodes are cooperative and trustworthy. Therefore, attackers can join the network and without any supervision they can easily manipulate the routing protocols and disrupt the routing process and thus the whole network activities.

**Lack of central administration:** In networks with a central administration there are places such as routers, switches, firewalls that a monitoring can be
placed and therefore a border can be defined for the network. Lack of central administration diminishes the sense of border for ad hoc networks, and there are no definite places that monitoring can be placed. Thus, attacker could execute their malicious activities with less chance of being detected.

**Limited resources:** Usually most of the nodes in ad hoc networks have limited resources. The main limitation usually is related to the battery power and then computation power, memory and bandwidth. In many different ways malicious nodes could take advantage of these limitation and disturb the network. For example, by making a service node busy, it can goes out of battery power and thus that service become unavailable to the network. Some routing algorithms and protocols tries to improve the network security by adding extra features in their process which these could lead to more consumption of computation power, memory, bandwidth and therefore energy [29].

### 4.2 Attacks

As it is mentioned earlier ad hoc networks are prone to the same attacks from the normal wired and wireless networks, due to technical features that ad hoc networks inherit from the normal networks. In addition, as discussed in the previous section ad hoc networks have their own specific characteristics that make them vulnerable against some specific type of attacks. Attacks in all type of networks can be classified based on the following areas [36]:

- Passive or active
- Internal or external
- Different protocol layer
- Stealthy or non-stealthy
- Cryptography or non-cryptography related

This classification of the attacks is visualized in the figure 4.1.
• **Passive vs. active attacks**: depend on the way that an attack affects data transmission between nodes the attack can be categorized into passive or active. Active attacks affect the transmission of the data by actively manipulating, modifying, fabricating or interrupting data communication inside a network. Jamming and Denial of Service (DoS) are some of the most famous examples of active attacks. On the other hand, passive attacks do not have any tangible effects on the transmitted data during the transmission. They just gather and monitor data for future malicious activities. The most famous passive attack is eavesdropping.

• **Internal vs. external attacks**: based on the domain of the attack they are either internal or external. While internal attacks are carried from insider nodes, the external attacks executed by nodes that do not belong to the network. Because the inside nodes usually carry sensitive information about the network the impact of the internal attacks could be more severe than the external attacks [36].

• **Attacks on different layers of the Internet model**: Attacks could take advantage of weaknesses and vulnerabilities that exist in different Internet model layers. Therefore, based on the layer that they took
advantage of they can be classified differently. However, some attacks might take advantage of weaknesses in more than one layer.

- **Stealthy vs. non-stealthy attacks**: If attackers try to hide themselves from the networks monitoring system the attack is called Stealthy. However, not all attacks could be hidden, and they are called non-stealthy. The most noticeable non-stealthy attack is DoS attack [36].

- **Cryptography vs. non-cryptography related attacks**: Cryptography attacks are the ones that take advantages of weaknesses and vulnerabilities in cryptographic system inside a network. They find weaknesses in the cryptographic elements such as code, cipher, cryptographic protocol or key management scheme. When an attack does not take advantages of the cryptographic weaknesses it will be classified as non-cryptography attack.

Table 4.1 shows attacks on ad hoc networks that are categorized based on their level of activities in different Internet model layers. The attacks on the network layer are more specific to the ad hoc networks. This is because these attacks are mainly on the network layer, and this is the main layer that differentiates ad hoc wireless networks from the other type of networks. Therefore, in the remaining of this chapter the focus will be on the attacks on the networks layer, and they will be discussed in more details.

### 4.2.1 Network layer attacks

Network layer is very critical to the ad hoc wireless networks. This is the layer that extends connectivity of nodes inside an ad hoc network from 1-hop to the whole network. This is where the main characteristics of ad hoc networks, infrastructure-less and self-administration, is defined. Attacks in this layer use weaknesses and vulnerabilities of the current routing protocols and execute their malicious activities. Most of them can be categorized in the followings [14]:

- **Routing Table Overflow**: These attacks by overflowing the routing table try to make the routing protocol dysfunctional. Therefore, they add as many routes as it is required to the routing table until no other route could be added to the table. The proactive routing algorithms are more vulnerable to this type of attacks than reactive routing algorithms. It is because proactive routing protocols attempt to discover routing information before it is actually needed. Therefore, malicious nodes can send excessive route advertisements to overflow their victims’ routing tables.
• **Routing Table Poisoning:** In this type of attack malicious nodes send fake or modified routing updates to the other nodes. When routing tables in the normal nodes poisoned, there would be either delay or missing packets in the network. Therefore, there could be a reduction in the networks performance or even some part of the network could become inaccessible.

• **Packet Replication:** In order to make network dysfunctional and reduce its performance, malicious nodes in this type of attacks broadcast stale packets. This could lead to wrong routing information or just additional consumption of bandwidth and power.

• **Route Cache Poisoning:** In this type attacker take advantage of the promiscuous mode of routing algorithm, where a node could add the routing information contained in packet headers that it overhears to its route cache. Malicious nodes send spoofed packet that hold a route to a specific destination via themselves. Other nodes that overhear those packets, add a route to that destination through the malicious node. The objective of this sort of attacks is same as the routing table overflow attacks.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application layer</td>
<td>Repudiation, data corruption</td>
</tr>
<tr>
<td>Transport layer</td>
<td>Session hijacking, SYN flooding</td>
</tr>
<tr>
<td>Network layer</td>
<td>Wormhole, black-hole, Byzantine, flooding, resource consumption, location disclosure attacks</td>
</tr>
<tr>
<td>Data link layer</td>
<td>Traffic analysis, monitoring, disruption MAC (802.11), WEP weakness</td>
</tr>
<tr>
<td>Physical layer</td>
<td>Jamming, interceptions, eavesdropping</td>
</tr>
<tr>
<td>Multi-layer attacks</td>
<td>DoS, impersonation, replay, man-in-the-middle</td>
</tr>
</tbody>
</table>

Table 4.1 - Attacks on ad hoc wireless networks based on their OSI layers [36]

After knowing the general approaches for attacks in the network layer of ad hoc wireless network, it would be good to look at some the most known attacks that compromise network security by misusing its protocols.
• **Wormhole attack**: An attacker or usually a pair of colluding nodes record packets at one location of a network and replay them at another location of the network. In this way they advertise shorter routes to a destination and they create a tunnel called wormhole tunnel [22]. As result, the routing algorithms gets disrupted and attackers try to collect, manipulate or just drop the packets that are passing through the tunnel. They are very dangerous attack in the MANET. For instance if this attack is used against an on-demand routing protocol such as DSR or AODV, the attack could prevent the discovery of any routes other than through the wormhole [14].

• **Black hole attack**: in this attack the malicious node advertises that it always has the best path to any destination (or particular destinations). After attracting other nodes attentions and pretending to establish a route to the destination, it drops all data packets that its receives. It is named after actual black holes because they both grab things and vanish them. All type of black hole attacks are discussed in detail in the chapter 5.

• **Byzantine attack**: This attack can be carried out by just one malicious node or a set of malicious nodes that work on collusion. Malicious nodes do different things such as creating routing loops, forwarding packets through non-optimal paths or selectively dropping packets, which results in disruption or degradation of the routing services [36].

• **Rushing Attack**: It is basically a denial-of-service attack that works on all the current on-demand routing protocols [11]. In current on-demand routing protocols only the first route discovery request is accepted and the rest will be suppressed. Attackers take advantage of this feature, and send a very fast felonious response to the route discoveries request; therefore, the other legitimate response will be dropped.

• **Resource consumption attack**: This is also known as the sleep deprivation attack. In this attack malicious nodes try to consume battery life of other nodes by requesting excessive route discovery or by forwarding unnecessary packets to the other nodes. This attack is in the category of packet replication.

• **Location disclosure attack**: Attackers gather and analyze location information of nodes in the network in order to reveals the structure of the network. They also try to identify identities of each node that involve in a communication, then by analyzing the traffic they try to find out the
traffic pattern of the network. They use all the gathered information to plan more sophisticated attacks in the future.
# 5 Black Hole Attack

Black hole attack is a very famous attack in wireless ad hoc networks and it is named after the actual black holes. In astronomy black hole is a place in space where gravity pulls so much that even light cannot get out [31]. Like a real black hole that sucks everything into itself and vanishes them, a malicious node in this type of attack tries to attract and receive all the packets in the network and drop them. Black hole attack is exclusive to ad hoc networks because of lack of central administration in the network.

In a typical black hole attack when a route request is generated in the network, the malicious node claims that it has the best (shortest) route to the destination. In this manner the malicious node receives the packets for a specific or general communications and it drops them all. The most common form of black hole attack is illustrated in the figure 5.1. However, depend on the existing routing protocol in the network the execution of the attack can be varied (this is discussed in detail in section 5.2). Moreover, there are some other attacks such as gray hole and selective forwarding that act almost the same, but they do not always drop all the packets that they receive.

The rest of this chapter discusses different classification of black hole attack and how it is executed under different routing protocols.

![Figure 5.1 – Black hole attack [33]](image)

Based on the routing protocol that an ad hoc network is using and number of attacker, black hole attack can be classified in different categories. In the following different categories of a black hole attack will be discussed.
5.1 Single and Collaborative (Co-operative) Black Hole Attacks

Some papers such as [36, 27, 34] divide black hole attack base on number of active malicious nodes in the attack. They classify the attack into Single and Collaborative (Co-operative) Back Hole.

*Single black hole* attack is when only one malicious node is active in the network or in specific region of the network.

*Collaborative or Co-operative black hole* attack is an attack where more than one malicious node is active in the network, and they coordinate their activities with each other. Collaboration among malicious nodes can increase their impact on the network and makes their detection more difficult.

5.2 Black Hole Attack in Different Routing Protocols (Proactive and Reactive)

A major part of a black hole attack is the way that it exploits the ad hoc routing protocol in the network, and advertises itself as a node that holds a valid route to a destination. Based on the type of the routing protocols, reactive and proactive, the attacker uses different approaches to perform the attack. These mechanisms are categorized in the followings.

5.2.1 Black Hole Attack in Reactive (on-demand) Routing Protocols

Based on the position of malicious node and the way the attack is implemented the black hole attack, in on-demand routing protocols, can be divided in the following categories:

- Internal and External
- RREQ attack and RREP attack

The following discuss each of these categories in more details.

5.2.1.1 Internal and External Black Hole Attacks

Based on the position of the malicious nodes in a black hole attack they can be divided into Internal and External [35].
Internal Black Hole Attack

In this type of black hole attack the malicious node is inside the network, and it fits inside the routes of given source and destination. The malicious node makes itself an active route as soon as it gets the chance, and disrupts the transmission by dismissing the data packets. The detection of the internal black hole attack could be harder because detection of misbehavior inside a network is difficult.

External Black Hole Attack

When the malicious node is physically located outside of the network it is called external black hole attack. External attacks are almost like the internal attacks. They just need some preliminary spoofing in order to get enough data to be able to infiltrate the network. When they infiltrate the network they become like internal attacks and their impact would be the same. The external black hole attack can be summarized in the followings [35]:

1. Malicious node detects the active route and notes the destination address
2. Malicious node sends a route reply packet (RREP) including the destination address field spoofed to an unknown destination address
3. Hop count value is set to lowest values and the sequence number is set to the highest value
4. Malicious node sends RREP to the nearest available node which belongs to the active route. This also can be sent directly to the source node if the route is available
5. The RREP received by the nearest available node will be relayed via the established inverse route to the source node
6. After receiving the RREP, the source node assumes it has the new data, and it will update its routing information
7. New route selected by source node
8. The malicious node is placed in the route and it will drop the received data

5.2.1.2 RREQ and RREP Black Hole Attacks

Black hole attacks in on-demand routing protocol can be classified into two categories: RREQ Black hole attack and RREP Black hole attack [9].

Since DSR and AODV have a very similar Route Discovery process (except the way they avoid the loops, DSR uses TTL and AODV uses Sequence Number), the following descriptions are based on the AODV in order to keep consistency and avoid duplication in explanations.

Black hole attack caused by RREP
As discussed in the chapter 3, in on-demand routing protocols when a node wants to communicate with any other node, it has to initiate a process called Route Discovery by broadcasting a RREQ message. It is because initially no node in the network has the information about the topology of the network. The attacker takes advantage of this topology unawareness. When a malicious node receives the RREQ, it immediately generates a RREP message with invalid information and sends it back to the source node. The RREP either contains a fake route to the desired destination through the malicious node or it has the route through a non-existent IP. The malicious RREP can be generated through following steps [9]:

- Set the type field to RREP (2)
- Set the hop count field to 1
- Set the originator IP address as the originating node of the route and the destination IP address as the destination node of the route
- Increase the destination sequence number by at least one
- Set the source IP address (in the IP header) to a non-existent IP address (Black hole)

Figure 5.2 - Black hole attack formed by fake RREP [9]

Figure 5.2 illustrate an example of black hole attack with RREP. In this example node S wants to send some data to node D. Because it has no available route to D it broadcasts RREQ, then node A receives the RREQ. Since node A does not have any route to D, it rebroadcasts the S’s RREQ. On the other hand, node M, which is malicious, immediately replies to S with a fake RREP. Later node S receives the actual response from node D through A. However, node M’s RREP holds a higher sequence number and less hop count; therefore, S assumes that the route provided by M is fresher route and choose it over the other. Then S starts its data transmission, and consequently the malicious node will drop all the data packets.

**Black hole attack caused by RREQ**

In this method the attacker generates a fake RREQ, which holds a non-existent node address, and pretends it is rebroadcasting a new RREQ from the source. The fake RREQ is produced in the following manner [9]:

![Diagram showing black hole attack](image-url)
Set the type field to RREQ (1)
Set the originator IP address to the originating node’s IP address
Set the destination IP address to the destination node’s IP address
Set the source IP address (in the IP header) to a non-existent IP address (Black hole)
Increase the source sequence number by at least one, or decrease the hop count to 1

Figure 5.3 - Black hole attack formed by fake RREQ [9]

Figure 5.3 illustrates an example of black hole attack with RREQ. In this example, node S is transmitting data to node D, and it has already found route for its transmission. During the transmission node M, which is malicious, broadcasts a fake RREQ, which will be received by A, and B. Because A and B receive it through a new node with a higher sequence number they assume that it is a new RREQ and think that node S looking for a new route. This reinitiates a new route discovery and turns the situation into the state that attacker can inject itself into the route by using fake RREP that explained earlier.

5.2.2 Black Hole Attack in Proactive (Table-driven) Routing Protocols

Although the general concept and results of black hole attack in proactive protocols is similar to the black hole attack in reactive protocols, the implementation of the attack is different. This is clearly because the proactive protocols work in different way form reactive protocols. Therefore, in this section different implementations of black hole attack in OLSR are discussed. OLSR is chosen because it is the most established proactive routing protocol in wireless ad hoc network; therefore, it is discussed and analyzed the most in academic literatures such as [6, 29, 17].

Black Hole Attack in OLSR

As it is discussed in the chapter 3, in OLSR routing protocol HELLO and TC messages are used for advertisement and establishment of valid routes. Also, as it is mentioned earlier in this chapter, the first part of the black hole attack is
advertising itself as the best route. Therefore, the malicious node in OLSR can modify either HELLO or TC messages or even both in order to advertise false routes. Consequently, there are three different ways that a black hole attack can be implemented in OLSR: TC-Black-Hole, HELLO-Black-Hole, and TC-HELLO-Black-Hole [28].

**TC-Black-Hole Attack:** Malicious node in this attack tries to advertise fake routes by modifying TC messages. The malicious node puts addresses of all nodes in the network in its TC messages and propagates them into the network. In this way the malicious node claims that all nodes in the network have selected it as their MPR. Therefore, network nodes update their topology tables by this fake information, and they will choose routes that would lead to the malicious node for their transmissions.

**HELLO-Black-Hole attack:**

This type of black hole attack is done by manipulation of HELLO messages. Malicious node inserts all the addresses of the network into its HELLO messages that it broadcasts. In this way the attacker claims that all of the nodes in the network are its neighbors and they have a bi-directional link with it. Therefore, its neighbors assume that the malicious node is a perfect candidate for their MPRs, and they will choose it as their only MPR. This is because the malicious node claimed that it covers all of the level-2 neighbors and OLSR always tries to minimize number of MPRs. Then this MPR selection will be propagated through the network. Therefore, any node that is not level-1 or level-2 neighbor of the malicious node’s neighbors will send its data through the malicious node and vice versa. In order to cover bigger part of the network more than one malicious node is required in this type of black hole attack.

**TC-HELLO-Black-Hole attack:** This black hole attack is a combination of both TC-Black-Hole and HELLO-Black-Hole attacks. In addition to modifying HELLO massages, the malicious node also modifies the TC messages as well. Therefore, similar to HELLO-Black-Hole malicious node would be selected by all of its neighbors as MPR, and also similar to TC-Black-Hole it would propagate much fake links using fake TC messages. As a result, the malicious node will be able to advertise more fake routes and attract more network traffic towards itself.
6 Implementation of Simulator

Now that wireless ad hoc networks are introduced and different type of black hole attacks are discussed, it is time to put all those theories into a test. A good way to test the behavior of black hole attack is simulation. Since there are no ready to use simulation environment for simulating malicious activities in wireless ad hoc networks, a simulator had to be built. This chapter explains why and how the code for these kinds of simulations is implemented.

6.1 Simulation Tool

As it is mentioned in the introduction of this chapter no ready to use simulation software exist for simulating malicious activities in a network. Therefore it has to be implemented manually by coding either from scratch or using some network simulators. MATLAB was chosen as the development environment for the simulation. The reason was laid further than the goal of this work. The long-term vision has been to implement an easy to develop framework that could be expanded for simulation of all sorts of malicious activities in wireless ad hoc networks. The more details about MATLAB are discussed in the following.

6.1.1 MATLAB

MATLAB® is a high-level language and interactive environment for numerical computation, visualization, and programming. It is developed by Mathworks. In its very begging it was designed for matrix manipulation and it is where its names come from MATrix LABoratory. It was initially developed in the late 1970s and since then it has gained more than one million users in industry and academia, and it has vast range of applications from signal processing to computational biology. [37]

There were three major reasons that MATLAB was picked up as environment for this project. First, as it is mentioned is a high –level language, which makes it quite easy to learn and work with. It has good tools for debugging and monitoring of the code, and no additional software, setup or knowledge is required. The learning curve for every one is quite short. Thus, its easiness makes MATLAB suitable choice for fast prototyping. The second reason was its popularity among the academia and industry. It is a well-known and popular application among researchers from different background, so the chance that everyone can understand and use it is high. Third, there has been a well-written open source code available that eliminates needs of starting from scratch. Having this base code provided the opportunity to spend more
time on the problems and simulation scenarios and less time on coding of primitive wireless communications.

6.2 Modeling of Network

Almost all the aspects of the network are customizable in this code. The main routing protocol that is used for the simulations of this work is DSR, and there are simulations with and without mobility in – considering different nodes' speed. Number of nodes and percentages of malicious nodes are variable in each scenario. In addition, there are three types of positions available for the topology of the network: random, grid and custom defined. Also, Constant Bit Rate (CBR) application is used for demonstration of data transmission in the network. Since each of these parameters is customized to the purpose of each simulation, their details and values are discussed in the next chapter.

6.3 Base Code

Wireless Network Simulator in MATLAB is an open source code for mobile wireless networks in MATLAB. It is licensed as Public Domain and hosted by Sourceforge, which can be found in the following address:

http://wireless-matlab.sourceforge.net/

This base code provides the following functionalities: [38]

- Radio propagation: free space, two-ray, and lognormal shadowing
- Mobility: random waypoint model
- Physical layer: SNR-based packet capture, broadcast, dynamic transmission rate and power
- MAC layer: IEEE 802.11 (CSMA/CA, virtual carrier sense, and RTS-CTS-DATA-ACK)
- Network layer: Dynamic Source Routing (DSR)
- Application layer: overlay routing protocols

From the mentioned mechanisms and algorithms in the above-mentioned list DSR, which is discussed in section 3.5.1, is the only part that is modified in this work. It is because black hole attack had to be implemented in the network layer. Since the other mechanisms and algorithms related to other layers are not modified in this work, their explanations is not presented in here. However, each of the listed algorithms and mechanisms provides basic functionalities of each layer that is required for wireless ad hoc networks. Therefore, these already implemented parts saved a large amount of time by eliminating the need for
writing everything from scratch for simulations in this work. Also, simplicity of
the base code makes addition of extra functionalities such as malicious behaviors
an easy task.

The only downside of this base code is that it is written in procedural
programming paradigm. It was developed in 2006, and in that time MATLAB did
not support object-oriented programing (OOP) paradigm. Therefore, advantages
of the OOP could not be achieved easily with this code. However, because the
advantages it provides outweigh its disadvantage, this code was chosen as the
base code for this work.

6.4 Added Code

While the base code provides basic wireless ad hoc functionalities, it does not
support any type of malicious activities. Therefore, series of changes are made in
the code in order to add malicious behaviors. Also, some parts of the base code
are restructured in order to add more readability into the code. These changes
could simplify future developments and prepare the code for future
implementation of other wireless ad hoc attacks.

In this work black hole attack is implemented, and its implementation is added
into the action.m file. This is because in the base code functionalities of each
layer are implemented in the action.m, and black hole is executed by
manipulation of network layer. Also, main processes for monitoring of the
simulation are implemented in this file. The following briefly introduces added
parts to the base code and explains how they are organized.

![Simulation Setup](image)

**Figure 6.1 – Implemented web GUI for input parameters**
Simulation Scenarios

The value for the parameters of each simulation is defined in the sim.m file. It holds all the variables that define a scenario of each simulation. Before each simulation these variables have to be manually assigned with a proper value. There are lots of parameters to be considered and their values should not conflict with each other; therefore, a graphical user interface is designed to facilitate the process of customizing a simulation. This GUI, figure 6.1, is a web based interface, and it is written in HTML, CSS and JavaScript, which can be easily called inside the MATLAB's built-in web browser.

Nodes

After defining total number of nodes in the Sim.m, all nodes will be created in createNodes.m. Each node has three properties: id, type and timer. Id field in each node acts as unique identifier of each node during the simulation. The type field identifies if a node is malicious or not. Timer keeps internal malicious activity time for each node. It is only used for a certain scenarios when malicious nodes change their behaviors in different period of time.

Services

Any services in the network is defined and initiated in the service.m file. It is called after createNodes that created all the nodes in the network. It picks a series of nodes as sinks and sources. The selection is done based on the value of random_pair. If random_pair holds zero, it means that sources and sinks will be selected sequentially; otherwise, they will be selected randomly.

Malicious nodes

After assigning services to the network it is time for assigning malicious nodes, which is done in createMal.m file. First nodes that are assigned as source nodes will be eliminated from malicious selection. Then based on the type of attack the malicious node will be selected and initialized randomly (if they are not manually predefined in the sim.m).

Monitoring and reporting

Clearly monitoring and reporting are critical for any simulation. They help to evaluate correctness of the simulations and provide information for analysis of the data. Because different type of reports is required for controlling the system and analysis the data, the reporting is distributed in the following files and each generates a different type of report:

- printResults.m (and printResultsToFile.m)
- printFinalResults.m (and printFinalResultsToFile.m)
- *reportToExcel.m*

`printResults.m` generates the outcomes of each round of simulation and print it on the screen in the MATLAB, and `printResultsToFile.m` prints them inside a file for future references. `printFinalResults.m` summarizes the results of all rounds of simulations and print them on the screen in MATLAB. `reportToExcel.m` prints the generated results in `printResults.m` and `printFinalResults.m` in an excel file.
7 Simulations and Results

For better understanding of black hole attack impact on wireless ad hoc network, series of simulations are done in this this work. These simulations and their results are discussed in this chapter. However, before discussing the simulations, basic metrics and parameters that are used for evaluation of those simulations are explained.

7.1 Performance Metrics

In order to evaluate and analyze how an attack changes performance of a network, some metrics are required. The following are the metrics that are chosen for evaluation of simulations in this work.

**Delay:** It is the average end-to-end delay of all the packets received in the MAC layers of all nodes in the network, and it is measured in second. Clearly when the delay is higher the performance is worse. The delay can be a critical factor especially when QoS (Quality of Service) is required in the network.

**Throughput:** Also known as network throughput. It is the rate of successful message delivery over a communication channel. It is measure in bits/sec, and calculated by dividing total number of received packets in the sink node over total number of sent packets by the source. The system throughput or aggregate throughput is the sum of the data rates that are delivered to all terminals in a network. Throughput is a good indicator that how well the network can utilize its sources and transfer data from one node to another.

**Goodput:** It is the application level throughput. Dropped packets, retransmitted packets and protocol overheads are excluded. The goodput is the amount of useful information that is delivered per second to the application layer protocol.

7.2 Evaluation of The Simulation

After implementing a brand new simulation environment, the first thing is to make sure that the simulator can be trusted and its results are accurate. Therefore, the code is tested with some scenarios that their results were known even without any simulation. The code was tested for all the defined scenarios in this work; however, there were two basic setups that were run initially for validating the simulator. Those setups are illustrated in the figure 7.1.
In the first test only three nodes were used. In the first run all the nodes were clear and there were no malicious node. Therefore, node D received all the transmitted packets from node S. After that the middle mode is turned to a malicious node. Because in this setup there are no other node to transmit the node S's packets, it is expected that node D receives no data packet from node S. The results were confirming, and node D received zero data packet from node S. This means that the malicious node does its job in dropping packets, but what about the other attribute of black hole attack which is attracting packets by faking routes. This is tested in the scenario that is illustrated in the figure 7.1 (b). In this test again node S tries to send some data to node D. Although, node C has the shortest path to the D, node A should be able to present itself as the best route. Like the first test this scenario were run with and without malicious node. In the first run were there was no malicious node, node C was chosen as the best route to D, and D received all the transmitted data. In the second run as expected node A got the attention of S, and node D received no data packet from S. While the number of data packets that are received by node D is indicator of success or failure in each test, all other gathered data are tested in this way.

Through these tests it is ensured that the black hole implementation is correct and simulator works as expected. Then the actual simulations were run that are discussed in the following.

### 7.3 Simulations Setups

In total, 800 simulations are done for this work. Eight major scenarios with 80 variations are developed. The following parameters where considered for definition of each scenario:

- Topology of the network: Grid or randomly positioned
- Mobility of nodes: from no mobility up to 10 m/s (0, 2, 5, 10 m/s)
These parameters generated eight major scenarios. In addition, each scenario had ten variations of malicious nodes (0% - 90%). The case with one hundred percent malicious nodes is omitted because it is unrealistic and if it happens, there is nothing from the original network left for validation. Then each sub-scenario simulated 10 times. This is because one way of getting more realistic information through simulation is repeating the experiment. This amount of simulations generated large enough data that provide good accuracy for analysis of each situation.

The simulation area is 100 x 100 square meters. For the scenarios with mobility random waypoint model is used. Dynamic source routing (DSR) is used for the routing protocol. Also, a traffic generator was developed to simulate constant bit rate (CBR) sources. The size of data payload is 8 bytes, and the pause time is 2 second. Five pairs of source and sink exist in each simulation where the first five nodes are sources and the last five nodes are selected as sink. The grid topology and position of sources and sinks are illustrated in figure 7.2.

**7.4 Results and Discussion**

In every communication first important thing is to make sure that transmission of data were successful. Packet Delivery Ratio is a good indicator to show how successful data are transferred in a network. Figure 7.3 shows the packet delivery ratios of four different scenarios:

- Grid topology with no mobility
- Grid topology with mobility
• Random topology with no mobility
• Random topology with mobility

Figure 7.3 – Packet delivery ratio of different scenarios. Top left: Grid topology without mobility, Top right: Grid topology with mobility, Down Left: Random topology without mobility, down right: Random Topology with mobility

As it can be seen, grid topology with no mobility suffers the most from black hole attack. Since sources and sinks are in opposite side of each other in the grid topology, malicious node will be located exactly in the middle of them, and malicious node can attract data when it is located in the middle of the sources and destinations.

Figure 7.4 – Average Packet Delivery Ratio of all simulated scenarios
Adding mobility or randomness to the topology decreases the impact of attack on packet delivery ratio but as the trend lines in the lower charts of figure 7.3 confirm the impact is still noticeable. Because sources and sinks get the chance to become one-hop neighbors of each other, the attack has lesser impact when nodes are mobile or when the topology is random. However, because these scenarios are very specific, analyzing the effects of the attack on the average of these scenarios can provide a more general understanding toward the attack. Figure 7.4 shows the average packet delivery ratio of all the mentioned scenarios. Applying black hole attack to the network decreases the success rate of data delivery logarithmically. Since the impact is logarithmic the very first few malicious nodes can radically decrease data delivery of a service in the network. Introducing only three nodes drops successful delivery of packets from almost 100 percent to 65 percent, and if only 30 percent of the nodes turned into malicious the success rate drops to 39 percent. This means when 30 percent of the network acting as black holes, successful delivery of information in the network is reduced into almost one third of its normal situation.

Although data delivery still decreases by introduction of any additional malicious node into the network, the first few can totally disturb any network. This means unlike some other attacks limiting number of malicious nodes is not very helpful in black hole attack; therefore, the malicious nodes has to be eliminated altogether or at least limited below 5 percent in order to have more than 90 percent of successful data delivery in the network.

Figure 7.5 - Throughput of different scenarios. Top left: Grid topology without mobility, Top right: Grid topology with mobility, Down Left: Random topology without mobility, down right: Random Topology with mobility
The next metric is throughput. Figure 7.5 shows the throughput for each mentioned scenarios and the figure 7.6 shows the average throughput for all scenarios. As it can be seen any additional malicious node has negative effect on the throughput, and it gradually decreases. Although it can be seen that the throughput is decreasing with any additional malicious node, it is not clear that how this decrease affect the performance of network services. Therefore, it is better to look at goodput of the network.

Figure 7.6 – Average Throughput of all simulated scenarios

Figure 7.7 – Goodput of different scenarios. Top left: Grid topology without mobility, Top right: Grid topology with mobility, Down Left: Random topology without mobility, down right: Random Topology with mobility
As it is explained in the metrics section 7.1, goodput is a better metric for understanding the actual transmitted information in a network. Thus, goodput can indicate more clearly than throughput that how the attack affects the network services. Figure 7.7 shows the changes in the goodput when number of malicious nodes increases in different scenarios and the figure 7.8 show the goodput of average. Like previous metrics topology and mobility can decrease or increase the impact of the attack on the network.

The goodput clearly shows what black hole attack did to the network communication and it confirms the packet delivery ratio results. Since each malicious node can attract a portion of transmitted data in a network, additional malicious nodes mean addition data loss and less goodput. Therefore, after a certain level malicious nodes cover the whole network and they do not let any data being transferred to any destinations.

![Figure 7.8 – Average Goodput of all simulated scenarios](image)

Last metric is end-to-end delay that is shown in figure 7.9. In general when number of malicious nodes increases the end-to-end delay decreases in the network. The fluctuation in the figure 7.9 is normal because it represents an average delay of different topologies where malicious node randomly positioned, and position of malicious nodes have major impact on end-to-end delay. By looking into definition of end-to-end delay and implementation of black hole attack the observed changes in the end-to-end delay can be explained. As it is discussed in the chapter 5, malicious nodes do not rebroadcast the RREQs and they reply with fake RREP immediately. In this way source node setups its route faster and it starts its data transmission sooner. Therefore, increasing number of malicious nodes reduce the average route setup delay of the network. Also, The end-to-end delay is the average delay of all packets in the network, which includes delays of data packets and routing packets. Therefore, additional
malicious nodes in the network decrease the end-to-end delay of the network by reducing the end-to-end delay of routing packets.

Figure 7.9 – Average end-to-end delay of all simulated scenarios
8 Conclusion and Future Work

Wireless ad hoc networks are relatively new type of networks that need no infrastructure or central administration to operate. They have attracted a lot of attention in recent years. Because of their decentralized behavior, they can provide network connectivity and communication in situations where previously was not possible. Since they inherited many aspects of wired and wireless networks they also inherited some of their vulnerabilities. In addition, this lack of requirement for infrastructures also gives them some extra vulnerabilities. Therefore, security is a big concern in this type of networks.

This work reviewed attacks that are specific to wireless ad hoc network, but the main focus was on black hole attack. First it was explained that how it works and how it can be implemented in different routing protocols. Later, series of scenarios were defined for simulation in order to analyze the impact of the black hole attack on wireless ad hoc networks.

Simulations showed that the black hole can drastically decrease data delivery in a network. Although, depends on mobility and topology of the network the impact of the black hole can be varied, the decrease in goodput and data delivery ratio metrics in every situation is noticeable. Furthermore, the impact of the number of malicious nodes in the network was demonstrated. The data delivery decreases by half when only about 30 percent of the network is malicious. Moreover, it is possible that data delivery drops to zero when about 70 percent of the network is controlled by malicious nodes. These data suggest that black hole attack is a serious attack and major steps have to be taken to insure the network against it.

8.1 Future work

Base on this work there are multiple options that can be considered for future works. One way would be defining specific scenarios for specific needs, and run simulations on them. For instance, for better understanding of consequences of black hole attack on smart roads, movement of vehicles in roads can be modeled, and then the impact of the attack can be analyzed. This analysis can provide better insight about the attack and help in better prevention of the attack. Ultimately this might lead to development of countermeasures against black hole attack. Another option is implementation of other wireless ad hoc routing protocol attacks. Then results from their simulation can provide a good reference for comparing performance of each protocol against black hole attack. Final possibility could be the implementation and analysis of other wireless ad
hoc attacks. This could help to prioritize preparation of the network against the attacks that have the most negative impacts on the network performance.
References


[34] Irshad Ullah, Shoaib Ur Rehman. "Analysis of Black Hole attack on MANETs Using different MANET routing protocols." School of Computing Blekinge Institute of Technology, Sweden, 201.


Appendices

Source Codes

Sim.m

```matlab
function sim4(varargin)

    if nargin < 2
        clear;
    end

    global inParam;
    inParam = nargin;

    global n node nodeList;
    global random_topo random_mal_pos random_pair random_mal_time;
    global nmal attack round;
    global Monitoring Send Receive start_time start_time0;
    global send_data send_rts send_cts send_ack;
    global send_size recv_size send_app_size recv_app_size;
    global send_net_data send_net_rreq send_net_rrep;
    global receive_net_data receive_net_rreq receive_net_rrep;
    global receive_data receive_rts receive_cts receive_ack;
    global receive_me_net_data receive_me_net_rreq receive_me_net_rrep;
    global receive_me_data receive_me_rts receive_me_cts receive_me_ack;
    global send_app receive_app;
    global figure_Number;
    global Event_list;
    global ss_Candidates sources sinks;
    global udp_counter;
    global print_to_file;
    global mobility interval nrepeat;
    global maxx maxy;
    global mon_pktTime mon_allTime mon_macTime mon_netTime mon_appTime
    mon_pktCounter1 mon_pktCounter2;
    global nPackets;
    global fixedTopo grid gridx maxspeed maxpause mobility_model;
    global UDP_intervals mal_act_timer;
    global fastSimulation;
    global codeVersion;
    global newLineCounter;

codeVersion = 'v2.0.2';
newLineCounter = 0; %keep printed results clean!
%% :param
%% :genParam
log_file = 'log_crosslayer_';
fastSimulation = 1; %change report behavior by skipping printing
details
% int_slot_time = 20*1e-6; % it takes 20 microseconds, 1 microsecond
% = 1e-6
max_time = 100;

% apptype = 'crosslayer_searching';
% apptype = 'dht_searching';
apptype = 'UDPecho_crosslayer';
```

51
```plaintext
% apptype = 'UDPecho_dht';

%random = 1; %control randomness in different Runs
random_topo = 1; % (1): random topology
random_pair = 0; % (1): random sources and sinks, (0):
normal[first and last n nodes], (-1): manual configuration
random_mal_pos = 1; % (1): reset random selection of malicious
nodes
random_mal_time = 0; % (1): random activation of malicious nodes

mal_act_timer = 100*1e-6; %define malicious activity interval,
mal_act_timer = 0 makes the malicious node constantly active
%in order to use this value
random_mal_timer should be 0 %if random mal time = 1 the
nodeList.timer(n) = -1 which it makes malicious activities random
%malicious node starts normal and
activate on every second phase
mal_act_timer = 0;

%% :topoParam
n = 30; %total number of nodes
pairs = 5; %defines number of sources and sinks pairs
sources = [1];%IMPORTANT: number of its elements should match pairs
sinks = [25];
nPackets = 10; %number of packets source sends, nPackets = 1
act as the original program
UDP_intervals = 1; %intervals for UDPecho server
rounds = 10; %number of simulations that have to be run
ntopo = 1; %default =2
nsize = 1; %default =2, but it should be kept as 1 since I
am suing different scenario

%% :malParam
attack = 1; %type of the attack. (0) no malisious activity /
(1) mal nodes just drop any packt / (2) mal nodes only drop pkt from
listed nodes
%(3) mal node cause collision
mStatus = 1; %0):fix[takes its value from nmal] / (1):variable
[in each rounds it takes its value from nmal_List] / (2): its value
is the length of mList
nmal = 0;
nmal_List = [0 3 6 9 12 15 18 21 24 27];
mList = [2 5]; %define malicious nodes when mStatus is 1 and it
override nmal_List
dropList = [1,2,3,4,5]; %list of nodes that malicious nodes drop
their pkts in attack=2

%% :fixtopo
fixedTopo = true;
customeTopo = 2; %(0):deactivate (1):user inout (2):predefined
grid = true;
gridx = 3; %1:16 (2):25 (3):30 (4):50 (5):100

%% :mobility
mobility = false; %(0):mobility is on / (0):mobility is off
```

52
nmobility = 1;
nrepeat = 1;
interval = 10;  % second
max_time = 100 + interval * (nrepeat + 1);

mobilityModel = 'random_waypoint';
max_pause = 1;
max_speed = 10;

%% :monParam
draw = 0;        % (0): no charts, (1): all charts, (2): just topo, send/recv and app_send/recv charts
print_to_file = 1; % (0): no charts, (1): all charts, (2): just topo, send/recv and app_send/recv charts
% ------------------End of Parameters-----

if nargin > 1  % GUI input
    n = varargin{1};
    pairs = varargin{2};
    nmal = varargin{3};
    attack = varargin{4};
    % dropList = varargin{};
    random_topo = varargin{5};
    random_pair = varargin{6};
    random_mal_pos = varargin{7};
    random_mal_time = varargin{8};
    rounds = varargin{9};
    nPackets = varargin{10};
    UDP_intervals = varargin{11};
    draw = varargin{12};
    mal_act_timer = varargin{13} * 1e-6;
    fixedTopo = varargin{14};
    grid = varargin{15};
    gridx = varargin{16};
    apptype = varargin{17};
    customeTopo = varargin{18};
    nmal_List = varargin{19};
    sources = varargin{20};
    sinks = varargin{21};
    mStatus = varargin{22};
    print_to_file = varargin{23};
    mobility = varargin{24};
end

Node = [];  
if customeTopo == 1
    fixedTopo = 1;
    figure,
    Node = ginput(n);
    Node = Node*70;
end

% if rounds ==1
%    nmal_List(1) = nmal;
% end

udp_counter = 1;

figure_Number = 1;  % keep track of the last figure
Send = [];          % keeps sent data for all nodes for every round
send_data = [];     
send_rts = [];
send_cts = []; send_ack = []; Receive = []; % keeps received data for all nodes for every round receive_data = []; receive_rts = []; receive_cts = []; receive_ack = []; Monitoring = []; % keeps list of all nodes for every round send_net_data = []; send_net_rreq = []; send_net_rrep = []; receive_net_data = []; receive_net_rreq = []; receive_net_rrep = []; receive_me_net_data = []; receive_me_net_rreq = []; receive_me_net_rrep = []; receive_me_data = []; receive_me_rts = []; receive_me_cts = []; receive_me_ack = []; send_app = []; receive_app = []; Event_list = []; 

if fixedTopo if grid % creates a grid topology switch gridx 
 case 1 
 for i=0:30:100
 Node = [Node;ones(4,1)*i [0:30:100]'];
n = 16;
 end
 case 2 
 for i=0:25:100
 Node = [Node;ones(5,1)*i [0:25:100]'];
n = 25;
 end
 case 3 
 for i=0:25:125
 Node = [Node;ones(5,1)*i [0:25:125]'];
n = 30;
 end
 case 4 
 for i=0:25:225
 Node = [Node;ones(5,1)*i [0:25:225]'];
n = 50;
 end
 case 5 
 for i=0:25:225
 Node = [Node;ones(10,1)*i [0:25:225]'];
n = 100;
 end 
 end
 elseif customeTopo == 2
 Node = [ [0 0];[0 25];[0 50];[50 0];[25 25];[25 50];[50 25];[50 50]]; 
end
end

54
mon_pktCounter2(1:rounds,1:n) = 0;
mon_pktTime(1:rounds,1:n) = 0;

%%% [end]------------------------

Parameters

% Initialize random number generator
rng('default');
if random_topo == 1 rng shuffle; end %generate different
topology on each execution
%[org] rand('state', 0);
%[org] randn('state', 0);
startTime0 = tic;

for iround=1:1:rounds
    for imobility = 1:nmobility
        round = iround;
        %keep track of
current round for indexes, %default: round = iround;

        if mStatus == 1
            nmal = nmal_List(iround); %assign number of malicious
            nodes in each round!
        elseif mStatus == 2
            nmal = length(mList);
        end

        %
        % Required check that number of the malicious nodes do not exceed
        % (total node - (sources+sinks))
        if (attack >0) && (nmal > n)
            fprintf(2,'Error: number of malicious nodes cannot exceed
            total number of the network nodes!
            
            %fprintf(2,'Error: number of sink and sources cannot
            exeed number of malicious nodes.
            
            return;
        end

        %:definitions
        ss_Candidates = ones(1,n); %Sinks and sources candidates
        if ~(random_pair == -1)
            sources = zeros(1,pairs);
            sinks = zeros(1,pairs);
        end

        Send = [Send; zeros(1,n)]; %reset sent data for new
        round, but keep the data from previous one
        send_data = [send_data; zeros(1,n)];
        send_rts = [send_rts; zeros(1,n)];
        send_cts = [send_cts; zeros(1,n)];
        send_ack = [send_ack; zeros(1,n)];
        send_size = 0;
        recv_size = 0;
        send_app_size = 0;
        recv_app_size = 0;
        Receive = [Receive; zeros(1,n)]; %same as Send
        receive_data = [receive_data; zeros(1,n)];
        receive_rts = [receive_rts; zeros(1,n)];
        receive_cts = [receive_cts; zeros(1,n)];
        receive_ack = [receive_ack; zeros(1,n)];
        send_net_data = [send_net_data; zeros(1,n)];
        send_net_rreq = [send_net_rreq; zeros(1,n)];
        send_net_rrep = [send_net_rrep; zeros(1,n)];
        receive_net_data = [receive_net_data; zeros(1,n)];
        receive_net_rreq = [receive_net_rreq; zeros(1,n)];
        receive_net_rrep = [receive_net_rrep; zeros(1,n)];
        receive_me_net_data = [receive_me_net_data; zeros(1,n)];
        receive_me_net_rreq = [receive_me_net_rreq; zeros(1,n)];
receive_me_net_rrep = [receive_me_net_rrep; zeros(1,n)];
receive_me_data = [receive_me_data; zeros(1,n)];
receive_me_rts = [receive_me_rts; zeros(1,n)];
receive_me_cts = [receive_me_cts; zeros(1,n)];
receive_me_ack = [receive_me_ack; zeros(1,n)];
send_app = [send_app; zeros(1,n)];
receive_app = [receive_app; zeros(1,n)];
mon_pktCounter1 = 0;
mon_allTime = 0;
mon_macTime = 0;
mon_netTime = 0;
mon_appTime = 0;

if fixedTopo
    maxx = max(Node(:,1));
    maxy = max(Node(:,2));
else
    maxx = sqrt(100*100*n/30); %change 100*100 for bigger space between nodes
    %maxx = 100;
    maxy = maxx;
end

%%
disp(['| ========================================== Round: ' num2str(around) ' =========================================== |']);
disp('');
disp(' ===== Network size = ' num2str(n) '  maxx = maxy = ' num2str(maxx) ' =====');
for itopo = 1:ntopo
    % Reset the parameters parameter;
    %% :initiation
    createNodes();
    %initial all nodes
    if random_pair == -1
        for r = sources
            nodeList(r).type = 2;
            ss_Candidates(r) = 0;
        end
        for r = sinks
            nodeList(r).type = 2;
            ss_Candidates(r) = 0;
        end
    end
    Event_list = struct('instant','','type','','node','','app','','net','','pkt','');
    Services(itopo,pairs,apptype,slot_time,nPackets); %define Sinks and Sources
    createMal(mStatus, nmal,mal_act_timer,dropList,mList);
    %generate mal nodes according to attack
    Monitoring = [Monitoring; nodeList];
    %record list of all node (mal and nor) for each round
    %[org] rand('state', itopo);
    %[org] randn('state', itopo);
    rng shuffle;
    % Generate a random network topology
    if ~fixedTopo
        node = topo(n, maxx, maxy);
        node = [node, zeros(n, 2)];
    end
else
    node = [Node, zeros(n, 2)];
end
if mobility == 1
    mobility_model = mobilityModel;
    maxpause = max_pause;
    maxspeed = max_speed;
    disp([' ===== Maximum speed = ' num2str(maxspeed) '  ']);
% Initialize and start mobility
position_init;
end
%% Run the simulation
startTime = tic;
tstart = clock;
ode
disp('       ');
disp([num2str(n)]);
disp([num2str(nmal)]);
disp([num2str(nPackets)]);
fprintf('Mobility: %d
', mobility);
fprintf('Max Speed: %d

', maxspeed);
for i = 1:1:n
    if nodeList(i).type == 1 fprintf(2, '%d ', i); end
end
fprintf(2, '
');
disp([num2str(sources)]);
disp([num2str(sinks)]);
run(Event_list', max_time, [log_file, num2str(n)]);
node
printResults(maxx,maxy,draw,round);
disp(sprintf('--- Network size= %d, Topology id=%d, Running time=%g
', n, itopo, etime(clock, tstart)));
end
end
printFinalResults(rounds);
end
createNode.m

function createNodes()
global n nodeList results;
global Send Receive;
for i = 1:1:n
    nodeList(i).id = i;
    nodeList(i).type = 0; % type(0): normal node /
    type(1):malicious node / type(2): source or sink
    nodeList(i).timer = 0; % for malicious nodes:
    timer(-1): random activity / timer(n): swtich its activity type every
    n macroseconds
createMal.m

function createMal(mStatus, m, mal_act_timer, dropList, mList)

global random_topo random_mal_pos random_mal_time;
global n malList nodeList ss_Candidates;
global blacklist attack;

i=m;
rng('default');
if random_topo == 1 || random_mal_pos == 1
    rng shuffle;
end

malList = zeros(1,n);

%% no attack
if attack == 0
    % just for clarity! otherwise this if can be removed.
    return;

%% mal nodes drop any pkt
elseif attack == 1
    if mStatus == 2
        for i=1:1:length(mList)
            j = mList(i);
            nodeList(j).type = 1;
            if random_mal_time == 1
                nodeList(j).timer = -1;
            else
                nodeList(j).timer = mal_act_timer;
            end
        end
    else
        while i ~= 0
            r = randi([1,n]);
            % check if it is not already chosen
            % if nodeList(r).type == 0 % ignores source, sinks and
            % previously selected malicious
            if nodeList(r).type == 0 || nodeList(r).type == 2
                % ignores previously selected malicious
                nodeList(r).type = 1;
                if random_mal_time == 1
                    nodeList(r).timer = -1;
                else
                    nodeList(r).timer = mal_act_timer;
                end
            end
            % ss_Candidates(r) = 0; % remove the node from Sink and
            Source candidates
            i = i-1;
        end
        malList(r) = 1;
    end

elseif attack == 2
    % convert dropList to program usable vertex
    blacklist = zeros(1,n);
    for k=1:1:length(dropList)
if dropList(k) > n
fprintf(2,'ERROR: black listed node cannot be chosen as malicious!\n');
else
    blacklist(dropList(k)) = 1;
end
end
blackList
while i ~= 0
    r = randi([1,n]);
    % chose nodes other than the ones in the dropList
    % also check if is not already chosen
    if blacklist(r) ~= 1 && nodeList(r).type == 0 % ignores source, sinks, previously selected malicious and nodes in the drop list
        nodeList(r).type = 1;
        if random_mal_time == 1
            nodeList(r).timer = -1;
        else
            nodeList(r).timer = mal_act_timer;
        end
        %ss_Candidates(r) = 0;
        i = i-1;
        malList(r) = 1;
    end
end
% mal nodes creat collision
elseif attack == 3
    % future work!
    return;
end
return;

Services.m

function Services(itopo,pairs,apptype,slot_time,nPackets)
global random_topo random_pair;
global n nodeList;
global ss_Candidates sources sinks;
global mobility interval nrepeat;
global Event_list;
i=pairs; % required number of sources and sinks
rng('default');
if random_topo == 1 rng shuffle; end
if random_pair == 1 % random
    counter = 1; % for counting events
    ss_counter = 1; % for counting sources and sinks
    while i ~= 0
        % make sure r1 and r2 are not equal!
        % r1 = randi([1,n/2]);
        r1 = randi([1,n]);
        if ss_Candidates(r1) == 1
            % r2 = randi([(n/2)+1,n]);
            r2 = randi([1,n]);
            % check if they are available (not chosen before and are not malicious)
            if ss_Candidates(r2) == 1
                for t=1:nPackets
                    Event_list(counter).instant =
1+100*j*slot_time+((t-1)*0.01); %FIX_ME: define proper interval time between each packet.

%Source, requesting node
Event_list(counter).type = 'send_app';
Event_list(counter).node = r1;

%Sink, requesting key!
Event_list(counter).app.type = apptype;
Event_list(counter).app.key = r2;

%traffic_id %FIX_ME: double check the value: counter or j or r1
Event_list(counter).app.id1 = counter;

%topo_id
Event_list(counter).app.route = [];
Event_list(counter).app.hopcount = 0;
Event_list(counter).net = [];
Event_list(counter).pkt = [];

counter = counter + 1;
end

end

end

end

elseif random_pair == 0 %normal

counter = 1;
for t=1:nPackets
for k=1:pairs

Event_list(counter).instant = 1000*k*slot_time+((t-1)*0.5);

% Requesting node
Event_list(counter).type = 'send_app';
Event_list(counter).node = k;

%Sink, requesting key!
Event_list(counter).app.type = apptype;
Event_list(counter).app.key = n+1-k;

%traffic_id
Event_list(counter).app.id1 = k;

%topo_id
Event_list(counter).app.id2 = itopo;

Event_list(counter).app.route = [];
Event_list(counter).app.hopcount = 0;
Event_list(counter).net = [];
Event_list(counter).pkt = [];

if t == 1

sources(k) = k;
% add selected source to the list of sources
sinks(k) = n+1-k;
% add selected sink to the list of sinks

nodeList(k).type = 2;
% prevent from being chosen as malicious

nodeList(n+1-k).type = 2;
% prevent from being malicious

end
end
end

ss_Candidates(r1) = 0;
% remove selected source from the future candidates

ss_Candidates(r2) = 0;
% remove selected sink from the future candidates

i = i-1;
ss_counter = ss_counter+1;
end
end
end

end
from being chosen as malicious
end
    counter = counter + 1;
end
end
else  %manual
    counter = 1;
    for t=1:nPackets
        for k=1:pairs
            Event_list(counter).instant = 1+50*k*slot_time+(t-1)*0.01;
            Event_list(counter).type = 'send_app';
            Event_list(counter).node = sources(k);
            %Source, requesting node
            Event_list(counter).app.type = apptype;
            Event_list(counter).app.key = sinks(k);  %Sink, requesting key!
            Event_list(counter).app.id1 = k;  %traffic_id
            Event_list(counter).app.id2 = itopo;  %topo_id
            Event_list(counter).app.route = [];
            Event_list(counter).app.hopcount = 0;
            Event_list(counter).net = [];
            Event_list(counter).pkt = [];
            counter = counter + 1;
        end
    end
end
return;

action.m

... case 'recv_net'
    t = event.instant;
    i = event.net.src;
    j = event.node;
    if bdebug, disp(['time ' num2str(t) ' recv_net @ node ' num2str(j)]); end

    % take care of TTL at network layer
    event.pkt.ttl = event.pkt.ttl - 1;
    if event.pkt.ttl < 0
        if bdebug, disp(['recv_net: TTL from node ' num2str(i) ' to ' num2str(j) ' is negative, drop the packet']); end
        return;
    end
    if j == i | j == event.pkt.tx
        % I myself sent this packet, no action
        return;
    end
    if cdebug, disp(['time ' num2str(t) 'node ' num2str(event.pkt.tx) ' -> node ' num2str(j) ' with type ' event.net.type]); end

    mon_netTime = mon_netTime + (t - event.pkt.t1);
    %:attack_implementation
    if attack > 0
        %mal nodes drop any pkt
        if attack == 1
            if nodeList(j).type == 1 && j == event.net.dst
                if strcmp(event.net.type,'data')
                    if nodeList(j).timer == -1
                        %malicious node
                        return;
                    end
                end
            end
        end
    end

61
luck = randi([1,100]);
    if luck > 50
        if (attack_debug == 1 ) disp(['node ' num2str(j) ' dropped a packet']);
            %drop
            return
    end
elseif nodeList(j).timer == 0
%malicious node is constantly active
    if (attack_debug == 1 ) disp(['node ' num2str(j) ' dropped a packet']);
        %drop
        return
else if mod(floor(t/nodeList(j).timer),2) == 1
%it starts normal and drop packets on every second phase
    if (attack_debug == 1 ) disp(['node ' num2str(j) ' dropped a packet']);
        %drop
        return
end
end
%mal nodes drop only pkt which are originated from blackList
elseif attack == 2
    if nodeList(j).type == 1
        if blackList(i) == 1 && strcmp(event.pkt.type, 'data')
            if (attack_debug == 1 ) disp(['node ' num2str(j) ' dropped a packet']);
                %drop
                return;
        end
    end
end

printResults.m

function printResults(maxx,maxy,draw,round)
global n nmal malList results nodeList node;
global sources sinks;
global attack;
global Send Receive startTime;
global send_data send_rts send_cts send_ack;
global send_size recv_size;
global send_net_data send_net_rreq send_net_rrep;
global receive_net_data receive_net_rreq receive_net_rrep;
global receive_data receive_rts receive_cts receive_ack;
global receive_me_net_data receive_me_net_rreq receive_me_net_rrep;
global receive_me_data receive_me_rts receive_me_cts receive_me_ACK;
global send_app receive_app;
global mon_pktTime mon_netTime mon_pktCounter1 mon_pktCounter2;
global figure_Number;
global random_topo;
global print_to_file;
global nPackets simTime;
global total_send total_recv totalCP_send totalCP_receive
totalCP_route_send totalCP_route_receive;
global total_net_data_send total_net_data_receive total_app_send
total_app_receive;
global fixedTopo grid gridx mobility maxspeed maxpause
mobility_model;
global fastSimulation;

%disable filename warning
warning off MATLAB:print:SavingToDifferentName;

totalSent = 0;
totalReceived = 0;
totalSentMal = 0;
totalReceivedMal = 0;

total_send = 0;
total_recv = 0;
totalCP_send = 0;
totalCP_receive = 0;
totalCP_route_send = 0;
totalCP_route_receive = 0;
total_net_data_send = 0;
total_net_data_receive = 0;
total_app_send = 0;
total_app_receive = 0;

for i = 1:1:n
    total_send = total_send + Send(round,i);
total_recv = total_recv + Receive(round,i);
totalCP_send = totalCP_send + send_rts(round,i) + send_cts(round,i) + send_ack(round,i);
totalCP_receive = totalCP_receive + receive_rts(round,i) + receive_cts(round,i) + receive_ack(round,i);
totalCP_route_send = totalCP_route_send + send_net_rreq(round,i) + send_net_rrep(round,i);
totalCP_route_receive = totalCP_route_receive + receive_net_rreq(round,i) + receive_net_rrep(round,i);
total_net_data_send = total_net_data_send + send_net_data(round,i);
total_net_data_receive = total_net_data_receive + receive_net_data(round,i);
total_app_send = total_app_send + send_app(round,i);
total_app_receive = total_app_receive + receive_app(round,i);
end

if fastSimulation == 0
    for i = 1:1:n
        if nodeList(i).type == 1
            disp( '-----------------' );
            fprintf(2,'Node %d
', i);
            fprintf(2,'Received: %d
', Receive(round,i));
            fprintf(2,'Sent: %d
', Send(round,i));
            totalSentMal = totalSentMal + Send(round,i);
totalReceivedMal = totalReceivedMal + Receive(round,i);
        else
            disp( '-----------------' );
            disp( [ 'Node ' num2str(i)]);
            disp( [ 'Received: ' num2str(Receive(round,i))]);
            disp( [ 'Sent: ' num2str(Send(round,i))]);
            totalSent = totalSent + Send(round,i);
totalReceived = totalReceived + Receive(round,i);
        end
    end
    if attack == 0 nmal = 0; end
    disp( '====================');
    disp( '       ');
disp( ['Total: ' num2str(n)]);
disp( ['Malicious: ' num2str(nmal)]);
disp( ['Packets: ' num2str(nPackets)]);
fprintf('Mobility: %d\n', mobility);
fprintf('Max Speed: %d\n', maxspeed);
disp( ['Average Sent (normal): ' num2str(totalSent/(n-nmal))]);
disp( ['Average Sent (malicious): ' num2str(totalSentMal/nmal)]);
disp( ['Average Received (normal): ' num2str(totalReceived/(n-nmal))]);
disp( ['Average Received (malicious): ' num2str(totalReceivedMal/nmal)]);
fprintf(2,'Malicious Nodes: ');
if attack == 0
  fprintf('NO Malicious node');
else
  for i = 1:n
    if nodeList(i).type == 1 fprintf(2,'%d ', i); end
  end
end
fprintf(2,'\n');
disp(['Sources: ' num2str(sources)]);
disp(['Sinks: ' num2str(sinks)]);
disp(' ');
disp(['Delay: ' num2str(mon_netTime/mon_pktCounter1)]);
disp(' ');
disp(['total send size : ' num2str(send_size)]);
disp(['total recv size : ' num2str(recv_size)]);
disp(' ');
disp(['total send : ' num2str(total_send)]);
disp(['total recv : ' num2str(total_recv)]);
disp(['total cp_send : ' num2str(totalCP_send)]);
disp(['total cp_receive : ' num2str(totalCP_receive)]);
disp(['total cp_route_send : ' num2str(totalCP_route_send)]);
disp(['total cp_route_receive : ' num2str(totalCP_route_receive)]);
disp(' ');
disp(['total_net_data_send : ' num2str(total_net_data_send)]);
disp(['total_net_data_receive : ' num2str(total_net_data_receive)]);
disp(['total_app_send : ' num2str(total_app_send)]);
disp(['total_app_receive : ' num2str(total_app_receive)]);
disp(' ');
simTime = toc(startTime);
disp(sprintf('Simulation time: %gs', simTime));
disp(' ');
disp(['] ============== End of Round: ' num2str(round) ' ============== | ']);
disp(' ');

fileName = ['figure-' num2str(round)];
if print_to_file ~= 0
  x = 1:n;
  y1 = [Receive(round,:)];
  y2 = [y1; [Send(round,:)]];
  h1 = figure(figure_Number);
  if draw == 0
set(h1,'visible','off');
end
subplot(1,3,1), h1;
rBar = bar(x,y2,'grouped');
title('Results');
xlabel('node');
ylabel('packets');
legend('Receive','Send');
set(rBar(1),'facecolor','c');
set(rBar(2),'facecolor','y');
if print_to_file ~= 0
    print('-dpng',fileName,h1);
end

set(gcf,'PaperUnits','inches','PaperPosition',[0 0 24 6])

%% detailed send_received_app
x = 1:1:n;
y1 = [send_app(round,:)];
y2 = [y1;receive_app(round,:)];
h9 = figure(figure_Number);
if draw == 0
    set(h9,'visible','off');
end
subplot(1,3,3), h9;
rBar = bar(x,y2,'grouped');
title('Sent & Received - app');
xlabel('node');
ylabel('packets');
legend('send','receive');
set(rBar(1),'facecolor','g');
set(rBar(2),'facecolor','b');
if print_to_file ~= 0
    print('-dpng',h9);
end

%% draw position of each node
r=0.016;
h2 = figure(figure_Number);
if draw == 0
    set(h2,'visible','off');
end
subplot(1,3,2), h2;
axis equal
hold off;
plot(NaN); % for clearing the subplot
box on;
axis([-20, maxx+20, -20, maxy+20]);
set(gca, 'XTick', []);
set(gca, 'YTick', []);
axis([0, maxx, 0, maxy]);
set(gca, 'XTick', [0; maxx]);
set(gca, 'YTick', [maxy]);
hold on;
for j=1:1:n
    ang=0:0.01:2*pi;
xp=r*cos(ang);
yp=r*sin(ang);
x=rand*100;
y=rand*100;
plot(node(j,1),node(j,2),'Color','white');
text(node(j,1),node(j,2),num2str(j),'
num2str(j,'FontUnits','pixels','FontSize',12,'FontWeight','light');
if nodeList(j).type == 0  
plot(node(j,1)+xp*1000,node(j,2)+yp*1000,'Color','green');  
elseif nodeList(j).type == 1  
plot(node(j,1)+xp*1000,node(j,2)+yp*1000,'Color','red');  
elseif nodeList(j).type == 2  
if find(sinks == j)  
% it is sink  
plot(node(j,1)+xp*1000,node(j,2)+yp*1000,'Color','black');  
else  
% it is source  
plot(node(j,1)+xp*1000,node(j,2)+yp*1000,'Color','blue');  
end  
end  
end  
grid on;  
xlabel('X');  
ylabel('Y');  
if print_to_file ~= 0  
print('-dpng',h2);  
end  
if print_to_file == 5  
%% detailed send  
x = 1:1:n;  
y = [send_data(round,:); send_rts(round,:); send_cts(round,:); send_ack(round,:)];  
h3 = figure(figure_Number);  
if draw == 0 || draw == 2  
set(h3,'visible','off');  
end  
subplot(3,3,4), h3;  
rBar = bar(x,y, 0.5, 'stack');  
title('Sent');  
xlabel('node');  
ylabel('packets');  
legend('data','rts','cts','ack');  
print('-dpng',fileName,h3);  
f0 = figure('visible','off');  
bar(x,y, 0.5, 'stack');  
title('Sent');  
xlabel('node');  
ylabel('packets');  
legend('data','rts','cts','ack');  
print('-dpng',f0);  
x = 1:1:n;  
y = [ receive_data(round,:); receive_rts(round,:); receive_cts(round,:); receive_ack(round,:)];  
h4 = figure(figure_Number);  
if draw == 0 || draw == 2  
set(h4,'visible','off');  
end  
subplot(3,3,7), h4;  
rBar = bar(x,y, 0.5, 'stack');  
title('Received');  
xlabel('node');  
ylabel('packets');  
legend('data','rts','cts','ack');  
print('-dpng',fileName,h4);  
f0 = figure('visible','off');
bar(x,y, 0.5, 'stack');
title('Received');
xlabel('node');
ylabel('packets');
legend('data', 'rts', 'cts', 'ack');
print('-dpng',f0);

%% detailed send_net
x = 1:1:n;

y = [send_net_data(round,:) send_net_rreq(round,:)]
send_net_rrep(round,:)];

h5 = figure(figure_Number);
if draw == 0 || draw == 2
    set(h5, 'visible', 'off');
end

dct = bar(x,y, 0.5, 'stack');
title('Send - net');
xlabel('node');
ylabel('packets');
legend('data', 'rreq', 'rrep');
print('-dpng', fileName, h5);
f0 = figure('visible', 'off');
bar(x,y, 0.5, 'stack');
title('Send - net');
xlabel('node');
ylabel('packets');
legend('data', 'rreq', 'rrep');
print('-dpng', f0);

%% detailed receive_net
x = 1:1:n;

y = [receive_net_data(round,:) receive_net_rreq(round,:)]
receive_net_rrep(round,:)];

h6 = figure(figure_Number);
if draw == 0 || draw == 2
    set(h6, 'visible', 'off');
end

rBar = bar(x,y, 0.5, 'stack');
title('Received - net');
xlabel('node');
ylabel('packets');
legend('data', 'rreq', 'rrep');
print('-dpng', fileName, h6);
f0 = figure('visible', 'off');
bar(x,y, 0.5, 'stack');
title('Received - net');
xlabel('node');
ylabel('packets');
legend('data', 'rreq', 'rrep');
print('-dpng', f0);

x = 1:1:n;
y = [ receive_me_data(round,:) receive_me_rts(round,:) ]
receive_me_cts(round,:)] receive_me_ack(round,:)';

h7 = figure(figure_Number);
if draw == 0 || draw == 2
    set(h7, 'visible', 'off');
end

subplot(3,3,6), h7;
rBar = bar(x,y, 0.5, 'stack');
function [ ] = printFinalResults(rounds)

    global n nmal mMalList results nodeList node;
    global ssCandidates sources sinks;
    global rv_threshold_delta;
    global attack;
    global Monitoring Send Receive startTime startTime0;
    global random_topo random_pair random_mal_time;
    global inParam;
    global nPackets;

disp('=================================================================================================================================================================================================================================================================================================================================================================================================');
disp('=================================================================================================================================================================================================================================================================================================================================================================================================');

printFinalResults.m
Report================================='

disp( ['Total : ' num2str(n)]);
disp( ['Malicious : ' num2str(nmal)]);
disp( ['Packets : ' num2str(nPackets)]);
disp( ['Attack type: ' num2str(attack)]);
disp( ['Rand_topo : ' num2str(random_topo)]);
disp( ['Rand_pair : ' num2str(random_pair)]);
disp( ['Rand_act : ' num2str(random_mal_time)]);
disp( ' ');
fprintf(2, 'Mal. Nodes : ');
if attack == 0
    fprintf('NO Malicious node');
else
    for i = 1:n
        if nodeList(i).type == 1 fprintf(2, '%d ', i);
    end
end
fprintf(2, 'n');
disp( ['Sources : ' num2str(sources)]);
disp( ['Sinks : ' num2str(sinks)]);
disp( ' ');
endTime = toc(startTime0);
disp(sprintf('Total simulation time: %gs', endTime));
disp(' ');
if attack == 0 nmal = 0; end
totalSent = 0;
totalReceived = 0;
totalSentMal = 0;
totalReceivedMal = 0;
for i=1:1:rounds
    for j = 1:n
        if Monitoring(i,j).type == 1
            sentMal(i) = sentMal(i) + Send(i,j);
            receivedMal(i) = receivedMal(i) + Receive(i,j);
        else
            sent(i) = sent(i) + Send(i,j);
            received(i) = received(i) + Receive(i,j);
        end
    end
    fprintf('Round %d - Avg Sent: %3.1f / Avg Rec: %3.1f / Avg Rec Mal: %3.1f / Avg Sent Mal: %3.1f
...', i,sent(i)/(n-nmal),received(i)/(n-nmal),receivedMal(i)/nmal,sentMal(i)/nmal);
    totalSent = totalSent + (sent(i)/(n-nmal));
    totalSentMal = totalSentMal + (sentMal(i)/nmal);
    totalReceived = totalReceived + (received(i)/(n-nmal));
    totalReceivedMal = totalReceivedMal + (receivedMal(i)/nmal);
end

disp(' ');
disp( ['All rounds average']);
disp( ['Average Sent (normal) : ' num2str(totalSent/rounds)]);
disp( ['Average Sent (malicious) : ' num2str(totalSentMal/rounds)]);
disp( ['Average Received (normal) : ' num2str(totalReceived/rounds)]);
disp( ['Average Received (malicious): ' num2str(totalReceivedMal/rounds)]);

printFinalResultsToFile(rounds);

%load results in a web page
if inParam > 2
    web([results.html'], '-noaddressbox');
end
return;

printToExcel.m

function reportToExcel( round )

global n nmal nodeList;
global sources sinks;
global attack;
global Monitoring Send Receive startTime;
global max_size_mac_body size_mac_header size_rts size_cts size_ack size_plcp;
global send_data send_rts send_cts send_ack;
global send_size recv_size;
global send_net_data send_net_rreq send_net_rrep;
global receive_data receive_rts receive_cts receive_ack;
global receive_me_data receive_me_rts receive_me_cts receive_me_ack;
global send_app receive_app send_app_size recv_app_size;
global mon_pktTime mon_allTime mon_macTime mon_netTime mon_appTime mon_pktCounter1 mon_pktCounter2;
global random_topo random_pair random_mal_pos random_mal_time;
global nPackets simTime;
global total_send total_recv totalCP_send totalCP_receive totalCP_route_send totalCP_route_receive;
global total_net_data_send total_net_data_receive total_app_send total_app_receive;
global fixedTopo grid gridx mobility maxspeed maxpause mobility_model;
global UDP_intervals mal_act_timer;
global codeVersion;

%disable filename warning
warning off MATLAB:print:SavingToDifferentName;

reportFile = fopen('report-excel.xls', 'a');
if reportFile == -1, error(['Cannot open log file for RREQ and RREP']); end;

f = dir('report-excel.xls');
if f.bytes == 0
    fprintf(reportFile,'n\t');
    fprintf(reportFile,'nmal\t');
    fprintf(reportFile,'attack\t');
    fprintf(reportFile,'nPackets\t');
    fprintf(reportFile,'fixedTopo\t');
    fprintf(reportFile,'grid\t');
    fprintf(reportFile,'gridx\t');
    fprintf(reportFile,'mobility\t');
    fprintf(reportFile,'speed\t');
    fprintf(reportFile,'pause\t');
fprintf(reportFile,"mobility_model\t");
fprintf(reportFile,"rnd_topo\t");
fprintf(reportFile,"rnd_pair\t");
fprintf(reportFile,"rnd_mal_pos\t");
fprintf(reportFile,"rnd_mal_time\t");
fprintf(reportFile,"UDP_intervals\t");
fprintf(reportFile,"mal_act_timer\t");
fprintf(reportFile,"version\t-\t");
fprintf(reportFile,"simTime (m)\t");
fprintf(reportFile,"simTime (s)\t-\t");
fprintf(reportFile,"total Pkt\t");
fprintf(reportFile,"pkts T_total\t");
fprintf(reportFile,"delay\t##\t");
fprintf(reportFile,"max_size_mac_body\t");
fprintf(reportFile,"size_mac_header\t");
fprintf(reportFile,"size_rts\t");
fprintf(reportFile,"size_cts\t");
fprintf(reportFile,"Size_send_total\t");
fprintf(reportFile,"Size_recv_total\t");
fprintf(reportFile,"Size_app_send_total\t");
fprintf(reportFile,"Size_app_recv_total\t");
fprintf(reportFile,"send_total\t");
fprintf(reportFile,"recv_total\t-\t");
fprintf(reportFile,"CP_send\t");
fprintf(reportFile,"CP_recv\t");
fprintf(reportFile,"CP_rout_send\t");
fprintf(reportFile,"CP_rout_recv\t-\t");
fprintf(reportFile,"data_send total\t");
fprintf(reportFile,"data_recv total\t-\t");
fprintf(reportFile,"app_send total\t");
fprintf(reportFile,"app_recv total\t-\t");
fprintf(reportFile,"OH_MAC\t");
fprintf(reportFile,"OH_NET\t");
fprintf(reportFile,"OH_TOTAL\t-\t");
fprintf(reportFile,"allPktsTransTime_Total\t");
fprintf(reportFile,"macPktsTransTime_Total\t");
fprintf(reportFile,"netPktsTransTime_Total\t");
fprintf(reportFile,"appPktsTransTime_Total\t##\t");
end

fprintf(reportFile,"\n%d\t", n);
fprintf(reportFile,"%d\t", nmal);
fprintf(reportFile,"%d\t", attack);
fprintf(reportFile,"%d\t", nPackets);
fprintf(reportFile,"%d\t", fixedTopo);
fprintf(reportFile,"%d\t", grid);
fprintf(reportFile,"%d\t", gridx);
fprintf(reportFile,"%d\t", mobility);
fprintf(reportFile,"%d\t", maxspeed);
fprintf(reportFile,"%d\t", maxpause);
fprintf(reportFile,"%s\t", mobility_model);
fprintf(reportFile,"%d\t", random_topo);
fprintf(reportFile,"%d\t", random_pair);
fprintf(reportFile,"%d\t", random_mal_pos);
fprintf(reportFile,"%d\t", random_mal_time);
fprintf(reportFile,"%d\t", UDP_intervals);
fprintf(reportFile,"%d\t", mal_act_timer);
fprintf(reportFile, "%s\t\t\t", codeVersion);
fprintf(reportFile, "%s\t\t\t", simTime/60);
fprintf(reportFile, "%s\t\t\t", simTime);
fprintf(reportFile, "%d\t\t\t", mon_pktCounter1);
fprintf(reportFile, "%d\t\t\t", mon_netTime);
fprintf(reportFile, "%d\t\#\#\t\t", mon_netTime/mon_pktCounter1);

fprintf(reportFile, "%d\t\t\t", max_size_mac_body);
fprintf(reportFile, "%d\t\t\t", size_mac_header);
fprintf(reportFile, "%d\t\t\t", size_rts);
fprintf(reportFile, "%d\t\t\t", size_cts);
fprintf(reportFile, "%d\t\t\t", send_size);
fprintf(reportFile, "%d\t\t\t", recv_size);
fprintf(reportFile, "%d\t\t\t", send_app_size);
fprintf(reportFile, "%d\t\t\t", recv_app_size);

fprintf(reportFile, "%d\t\t\t", total_send);
fprintf(reportFile, "%d\t\t\t", total_recv);
fprintf(reportFile, "%d\t\t\t", totalCP_send);
fprintf(reportFile, "%d\t\t\t", totalCP_receive);
fprintf(reportFile, "%d\t\t\t", totalCP_route_send);
fprintf(reportFile, "%d\t\t\t", totalCP_route_receive);
fprintf(reportFile, "%d\t\t\t", total_net_data_send);
fprintf(reportFile, "%d\t\t\t", total_net_data_receive);

fprintf(reportFile, "%d\t\t\t", total_app_send);
fprintf(reportFile, "%d\t\t\t", total_app_receive);
fprintf(reportFile, "%d\t\t\t", totalCP_send/total_app_receive);
fprintf(reportFile, "%d\t\t\t", totalCP_route_send/total_app_receive);
fprintf(reportFile, "%d\t\t\t", (totalCP_send + totalCP_route_send)/total_app_receive);

fprintf(reportFile, "%d\t\t\t", mon_allTime);
fprintf(reportFile, "%d\t\t\t", mon_macTime);
fprintf(reportFile, "%d\t\t\t", mon_netTime);
fprintf(reportFile, "%d\t\#\#\t\t", mon_appTime);

for i = 1:1:n
     fprintf(reportFile, '<%d>', i);
     fprintf(reportFile, '%d\t\t\t', send_data(round,i));
     fprintf(reportFile, '%d\t\t\t', send_rts(round,i));
     fprintf(reportFile, '%d\t\t\t', send_cts(round,i));
     fprintf(reportFile, '%d\t\t\t', send_ack(round,i));
     fprintf(reportFile, '%d\t\t\t', receive_data(round,i), receive_me_data(round,i));
     fprintf(reportFile, '%d\t\t\t', receive_rts(round,i), receive_me_rts(round,i));
     fprintf(reportFile, '%d\t\t\t', receive_cts(round,i), receive_me_cts(round,i));
     fprintf(reportFile, '%d\t\t\t', receive_ack(round,i), receive_me_ack(round,i));
     fprintf(reportFile, '%d\t\t\t', receive_net_data(round,i), receive_me_net_data(round,i));

     fprintf(reportFile, '%d\t\t\t', send_net_rreq(round,i));
     fprintf(reportFile, '%d\t\t\t', send_net_rrep(round,i));
     fprintf(reportFile, '%d\t\t\t', receive_net_rreq(round,i));
     fprintf(reportFile, '%d\t\t\t', receive_net_rrep(round,i));

     fprintf(reportFile, '%d\t\t\t', send_net_data(round,i));

}
fprintf(reportFile,'%d\t', receive_net_data(round,i));
fprintf(reportFile,'%d\t', send_app(round,i));
fprintf(reportFile,'%d\t', receive_app(round,i));
end

%malicious nodes
fprintf(reportFile,‘##’);
if attack == 0
    fprintf(reportFile,‘\t0’);
else
    for i = 1:n
        if nodeList(i).type == 1 fprintf(reportFile,‘\t%d’, i); end
    end
end
end