**University of Piraeus**

Department of Digital Systems

Systems Security Laboratory

**Master Thesis**

# Attacks in Wireless Sensor Networks

**Christina Skouloudi**

**September 2014**

## Table of Contents

## Table of Figures

*To Christina, Eftychia and Yianna*
*for their support and togetherness.*

*To Eleftheria*
*for helping me know myself.*

# Abstract

Wireless Sensor Networks have been studied in depth for several decades and entire books are devoted to the subject. Their main role is to provide bridges between the virtual world of information technology and the real physical world. They represent a fundamental paradigm shift from traditional inter-human personal communications to autonomous inter-device communications. They promise unprecedented new abilities to observe and understand large-scale, real-world phenomena at a fine spatio-temporal resolution. As a result, Wireless Sensor Networks also have the potential to engender new breakthrough scientific advances.

In this paper, our objective is to simulate and analyze the performance of routing protocols for Wireless Sensor Networks using a scenario based experiment in order to observe the network's behavior under all the simulated attacks. Moreover, it is examined through the simulation results if there is a pattern in all these attacks that could be used to detect or prevent a set of attacks.

**Keywords:** security, AODV, protocol, routing, mobile ad hoc network, Ad hoc, attack, ns2 simulator, metrics, blackhole, flooding, rushing, selective forwarding, packet delivery ratio, network throughput, average end to end delay, packet loss.

# Acknowledgements

Apart from the efforts of myself, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who contributed  in the successful completion of this extensive research.

I would like to show my greatest appreciation to Prof. Sokratis Katsikas who has the attitude and the substance of a genius. He continually and convincingly conveyed a spirit of adventure in regard to research and scholarship, and an excitement in teaching. He inspired me to become a wiser person.

I also want to thank Eleni Darra who introduced me to wireless sensor network attacks and provided me her support and continuous guidance on my master's thesis.

Finally, I thank my family for being always there for me. Without their economic support, their belief in me and their caring love I wouldn't be able to acquire the academic knowledge I have so far.

# Chapter 1

# Introduction to Wireless Sensor Networks

## 1.1 Definition

Wireless Sensor Networks (WSN) consist of a number of sensors which monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location. The “modern” networks are bi-directional, also enabling control of sensor activity. Common functions of WSNs are including broadcast and multicast, routing, forwarding and route maintenance. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance. Nowadays, such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on. WSNs are still vulnerable to many types of threats [1,2].

The WSN is built of "nodes" – from a few to several hundreds or even thousands, where each node is connected to one (or sometimes several) sensor. Each such sensor network node has typically several parts: a) a radio transceiver with an internal antenna or connection to an external antenna, b) a microcontroller, c) an electronic circuit for interfacing with the sensors and d) an energy source, usually a battery or an embedded form of energy harvesting. A sensor node might vary in size from that of a shoebox down to the size of a grain of dust, although functioning "motes" of genuine microscopic dimensions have yet to be created. The cost of sensor nodes is similarly variable, ranging from a few to hundreds of dollars, depending on the complexity of the individual sensor nodes. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and communication’s bandwidth. The topology of the WSNs can vary from a simple star network to an advanced multi-hop wireless mesh network. The propagation technique between the hops of the network can be routing or flooding [2].

If a centralized architecture is used in a sensor network and the central node fails, then the entire network will collapse, however the reliability of the sensor network can be increased by using distributed control architecture. Distributed control is used in WSNs for the following reasons:

- Sensor nodes are prone to failure
- For better collection of data
- To provide nodes with backup in case of failure of the central node

There is also no centralized body to allocate the resources and they have to be self-organized [2]. Depending on the application, WSN devices can be networked together in a number of ways. In basic data-gathering applications, for instance, there is a node referred to as the sink to which all data from source sensor nodes are directed. The simplest logical topology for communication of gathered data is a single-hop star topology, where all nodes send their data directly to the sink. In networks with lower transmit power settings or where nodes are deployed over a large area, a multi-hop tree structure may be used for data-gathering. In this case, some nodes may act both as sources themselves, as well as routers for other sources.

One interesting characteristic of wireless sensor networks is that they often allow for the possibility of intelligent in-network processing. Intermediate nodes along the path do not act merely as packet forwarders, but may also examine and process the content of the packets going through them. This is often done for the purpose of data compression or for signal processing to improve the quality of the collected information.

The main characteristics of a WSN include:

- Power consumption constraints for nodes using batteries or energy harvesting
- Ability to cope with node failures
- Mobility of nodes
- Communication failures
- Heterogeneity of nodes
- Scalability to large scale of deployment
- Ability to withstand harsh environmental conditions
- Ease of use
- Cross-layer design

## 1.2 Benefits, Constrains and Applications of WSNs

The advantages of a WSN are that it can be deployed easily, its structure is inexpensive as well as it can access and measure unreachable events. Also, it is a self-organized and self-managed network without the continuous need of a centralized management. The absence of wires accommodates its mobile movement too. However, WSNs have some design constrains such as their limited battery life, the limited instruction set and a very strict energy management. The fact of the unattended nodes also increases the challenge that a WSN faces.

According to [1], WSNs are vulnerable to many kinds of attacks not only because of the previous constrains but also because of insider attacks, reengineering, compromising and replicating of nodes, inapplicable/unusable traditional security techniques due to limited devices/resources, unreliable communications, deploy in open and hostile environments and interaction with physical environment.

In general, there are two kinds of applications for WSNs: **monitoring** and **tracking**. Therefore, some of the most common applications of these networks are:

- Military applications
- Environmental monitoring
- Health applications
- Home applications
- Smart cities
- Commercial applications
- Infrastructure protection
- Disaster detection and recovery
- Agriculture
- Law enforcement
- Transportation
- Space discovery
- Underground Mining
- Traffic Control
- Pipeline Monitoring

## 1.3 Node architecture

The architecture of a wireless sensor node, as shown in Figure 1, is descripted in detail in [2]. .

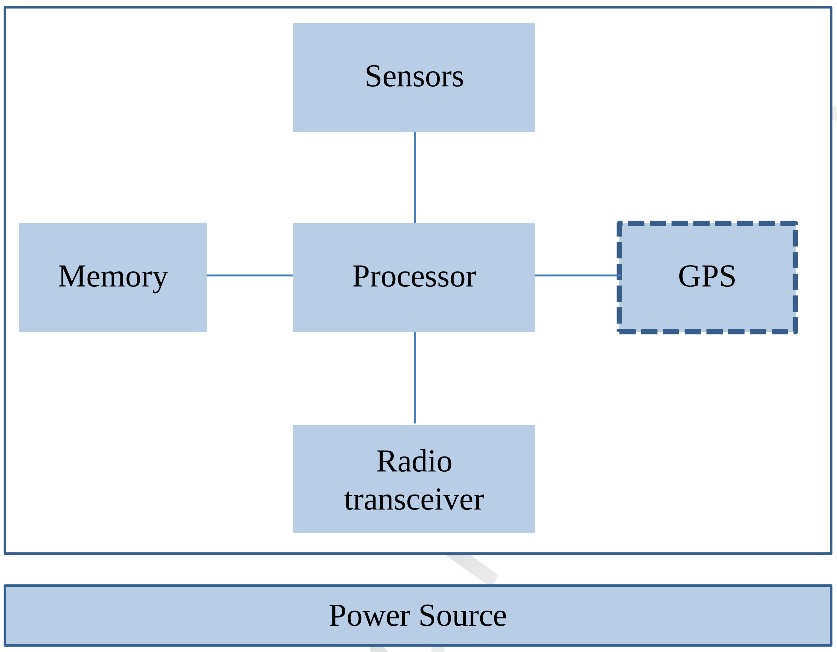

**Figure 1: WSN Node architecture**

### 1.3.1 Low-power embedded processor

The computational tasks on a WSN device include the processing of both locally sensed information as well as information communicated by other sensors. At present, primarily due to economic constraints, the embedded processors are often significantly constrained in terms of computational power (e.g., many of the devices used currently in research and development have only an eight-bit 16-MHz processor). Due to the constraints of such processors, devices typically run specialized component-based embedded operating systems, such as TinyOS. However, it should be kept in mind that a sensor network may be heterogeneous and includes at least some nodes with significantly greater computational power.

Moreover, future WSN devices may possess extremely powerful embedded processors. They will also incorporate advanced low-power design techniques, such as efficient sleep modes and dynamic voltage scaling to provide significant energy savings.

### 1.3.2 Memory/storage

Storage in the form of random access and read-only memory includes both program memory (from which instructions are executed by the processor), and data memory (for storing raw and processed sensor measurements and other local information). The quantities of memory and storage on board a WSN device are often limited primarily by economic considerations, and are also likely to improve over time.

### 1.3.3 Radio transceiver

WSN devices include a low-rate, short-range wireless radio (10–100 kbps, <100 m). While currently quite limited in capability too, these radios are likely to improve in sophistication over time – including improvements in cost, spectral efficiency, tunability, and immunity to noise, fading, and interference. Radio communication is often the most power-intensive operation in a WSN device, and hence the radio must incorporate energy-efficient sleep and wake-up modes.

### 1.3.4 Sensors

Due to bandwidth and power constraints, WSN devices primarily support only low-data-rate sensing. Many applications call for multi-modal sensing, so each device may have several sensors on board. The specific sensors used are highly dependent on the application; for example, they may include temperature sensors, light sensors, humidity sensors, pressure sensors, accelerometers, magnetometers, chemical sensors, acoustic sensors, or even low-resolution imagers.

### 1.3.5 Geopositioning system

In many WSN applications, it is important for all sensor measurements to be location stamped. The simplest way to obtain positioning is to pre-configure sensor locations at deployment, but this may only be feasible in limited deployments. Particularly for outdoor operations, when the network is deployed in an ad hoc manner, such information is most easily obtained via satellite-based GPS. However, even in such applications, only a fraction of the nodes may be equipped with GPS capability, due to environmental and economic constraints. In this case, other nodes must obtain their locations indirectly through network localization algorithms.

## 1.3.6 Power source

For flexible deployment the WSN device is likely to be battery powered (e.g. using LiMH AA batteries). While some of the nodes may be wired to a continuous power source in some applications, and energy harvesting techniques may provide a degree of energy renewal in some cases, the finite battery energy is likely to be the most critical resource bottleneck in most WSN applications.

Depending on the application, WSN devices can be networked together in a number of ways. In basic data-gathering applications, for instance, there is a node referred to as the sink to which all data from source sensor nodes are directed. The simplest logical topology for communication of gathered data is a single-hop star topology, where all nodes send their data directly to the sink. In networks with lower transmit power settings or where nodes are deployed over a large area, a multi-hop tree structure may be used for data-gathering. In this case, some nodes may act both as sources themselves, as well as routers for other sources [2].

# Chapter 2

# Security in Wireless Sensor Networks

The nature of large, ad-hoc, wireless sensor networks presents significant challenges in designing security schemes. A wireless sensor network is a special network which has many constraint compared to a traditional computer network because of its wireless medium, ad-hoc deployment, hostile environment, resource scarcity and immense scale, unreliable communication and the unattended operation [3]. There are many security services on WSNs; but some of their commons are including encryption and data link layer authentication, multi-path routing, identity verification, bidirectional link verification and authenticated broadcasts.

## 2.1 Security Goals for WSN

As the sensor networks can also operate in an ad-hoc manner the security goals cover both those of the traditional networks and goals suited to the unique constraints of ad-hoc sensor networks. The security goals are classified as primary and secondary. The primary goals are known as standard security goals such as Confidentiality, Integrity, Authentication and Availability (CIAA). The secondary goals are Data Freshness, Self-Organization, Time Synchronization and Secure Localization [3].

## 2.2 Security Attacks in WSN

A variety of attacks is possible in Wireless Sensor Network. These security attacks can be classified according to different criteria, such as the domain of the attackers, the OSI layer the attacks are applied [4,10], the techniques used in attacks or how the attacks are initially launched [32]. These security attacks in WSN and all other networks can be roughly classified by the following criteria: passive or active, internal or external, different protocol layer, stealthy or non-stealthy, cryptography or non-cryptography related. Here, attacks classification follows the structure of the OSI layer taxonomy.

### 2.2.1 Physical Layer Attacks

#### 2.2.1.1 Physical attack

Unlike many other attacks, physical attacks destroy sensors permanently, so the losses are irreversible. For instance, attackers can extract cryptographic secrets, tamper with the associated circuitry, modify programming in the sensors, or replace them with malicious sensors under the control of the attacker.

#### 2.2.1.2 Jamming

Jamming is the type of attack which interferes with the radio frequencies used by network nodes. It is an attack on physical layer of wireless network. It interferes with the radio frequencies being used by the nodes of a network. An attacker sequentially transmits over the wireless network refusing the underlying MAC protocol. Jamming can interrupt the network impressive if a single frequency is used throughout the network. In addition, jamming can cause excessive energy consumption at a node by injecting impertinent packets. The receiver's nodes will as well consume energy by getting those packets [5].

#### 2.2.1.3 Tampering

Another physical layer attack is tampering. Given physical access to a node, an attacker can extract sensitive information such as cryptographic keys or other data on the node. The node may also be altered or replaced to create a compromised node which the attacker controls. One defense to this attack involves tamper-proofing the node's physical package. However, it is usually assumed that the sensor nodes are not tamper-proofed in WSNs due to the additional cost. This indicates that a security scheme must consider [5].

#### 2.2.1.4 Radio interference

In Radio interference attack the adversary either produces large amounts of interference intermittently or persistently. To handle this issue, use of symmetric key algorithms in which the disclosure of the keys is delayed by some time interval [11].

#### 2.2.1.5 False Node

A false node involves the addition of a node by an adversary and causes the injection of malicious data. An intruder might add a node to the system that feeds false data or prevents the passage of true data. Insertion of malicious node is one of the most dangerous attacks that can occur. Malicious code injected in the network could spread to all nodes, potentially destroying the whole network, or even worse, taking over the network on behalf of an adversary [3].

### 2.2.2 MAC Layer Attacks

#### 2.2.2.1 Monitor and Eavesdropping

This is the most common attack to privacy. By snooping to the data, the adversary can easily discover the communication contents. When the traffic conveys the control information about the sensor network configuration, which contains potentially more detailed information than accessible through the location server, the eavesdropping can act effectively against the privacy protection [3].

#### 2.2.2.2 Man-in-the-Middle Attack

The man-in-the-middle attack is a form of active eavesdropping in which the attacker makes independent connections with the victims and relays messages between them, making them believe that they are talking directly to each other over a private connection. The attacker will be able to intercept all messages exchanging between the two victims and inject new ones [5].

#### 2.2.2.3 Traffic Analysis

Even when the messages transferred are encrypted, it still leaves a high possibility analysis of the communication patterns. Sensor activities can potentially reveal enough information to enable an adversary to cause malicious harm to the sensor network [3].

#### 2.2.2.4 Camouflage Adversaries

One can insert a node or compromise the nodes to hide in the sensor network. After that these nodes can copy as a normal node to attract the packets, then misroute the packets, conducting the privacy analysis [3].

#### 2.2.2.5 Location disclosure attack

An attacker reveals information regarding the location of nodes or the structure of the network. It gathers the node location information, such as a route map, and then plans further attack scenarios. Traffic analysis, one of the subtlest security attacks against WSN, is unsolved. Adversaries try to figure out the identities of communication parties and analyze traffic to learn the network traffic pattern and track changes in the traffic pattern. The leakage of such information is devastating in security-sensitive scenarios [36].

#### 2.2.2.6 Node Outage

Node outage is the situation that occurs when a node stops its function. In the case where a cluster leader stops functioning, the sensor network protocols should be robust enough to mitigate the effects of node outages by providing an alternate route [3].

#### 2.2.2.7 Node Malfunction

In this type of attack the compromised nodes behave normally but report false readings to lead to an incorrect decision. The malfunctioning node will generate inaccurate data that will expose the integrity of sensor network especially if it is a data-aggregating node such as a cluster leader [3].

#### 2.2.2.8 Collision

This is a DoS attack, where a node induces a collision in some small part of a transmitted packet. The packet will then fail the checksum check, because of the changes brought on by the collision, and the receiver node will then ask for a retransmission of the packet [4].

#### 2.2.2.9 Node Subversion

Capture of a node may reveal its information including disclosure of cryptographic keys and thus compromise the whole sensor network. A particular sensor might be captured, and information (key) stored on it might be obtained by an adversary [3].

#### 2.2.2.10 Misdirection

In this attack a malicious node, that is part of a route, can instead of dropping packets, quite simply send them on a different path where there does not exist a route to the destination. The malicious node can do this for certain packets, or all packets [4].

#### 2.2.2.11 Passive Information Gathering

An adversary with powerful resources can collect information from the sensor networks if it is not encrypted. An intruder with an appropriately powerful receiver and well-designed antenna can easily pick off the data stream. Interception of the messages, containing the physical locations of sensor nodes, allows an attacker to locate the nodes and destroy them. Besides the locations of sensor nodes, an adversary can observe the application specific content of messages including message IDs, timestamps and other fields. To minimize the threats of passive information gathering, strong encryption techniques needs to be used [3].

#### 2.2.2.12 Path based DoS

An adversary overwhelms sensor nodes by flooding a multi-hope end to end communication path with either replayed or injected false message to injected false message to waste secure energy resources [3].

#### 2.2.2.13 Clone Attack

Clone attack also known as node replication attack, is a severe attack in WSNs. In this attack, an adversary (WSN Adversary can be person or another entity that only monitors the communication channels which threatens the confidentiality of data) captures a few of nodes, replicates them and then deploys arbitrary number of replicas throughout the network. A node replicated in this approach can severely disrupt a sensor network’s performance. Packets can be corrupted or even misrouted by inserting the replicated nodes at specific network points. The attacker could easily manipulate a specific segment of the network, perhaps by disconnecting it altogether [3]. If an attacker can gain physical access to the entire network he can copy cryptographic keys to the replicated sensor nodes. In clone attack, an adversary may capture a sensor node and copy the cryptographic information to another node known as cloned node. Then this cloned sensor node can be installed to capture the information of the network. The adversary can also inject false information, or manipulate the information passing through cloned nodes [5].

#### 2.2.2.14 Resource consumption attack

This is also known as the sleep deprivation attack. An attacker or a compromised node can attempt to consume battery life by requesting excessive route discovery, or by forwarding unnecessary packets to the victim node [3,10].

### 2.2.3 Network Layer Attacks

The majority of these attacks take place at routing phase, routing discovery phase or routing maintenance phase.

#### 2.2.3.1 Sinkhole Attack

In Sinkhole attack, attacker attracts all the traffic from a particular area to a compromise node by advertising a zero cost route through itself. This is specified as a DOS attack. [4].

#### 2.2.3.2 Sybil Attack

In Sybil attack, a malicious node can represent multiple identities to the network. The Sybil attack targets fault tolerant schemes such as distributed storage, disparity, multipath routing and topology maintenance. This is done by having a malicious node present multiple identities to the network. This attack is especially confusing to geographic routing protocols as the adversary appears to be in multiple locations at once [4].

#### 2.2.3.3 Wormhole Attack

The simplest form of this attack is an attacker sits in between the two nodes and forward in between them. A malicious node uses a path outside the network to route messages to another compromised node at some other location [24]. In wormhole attack is not compromised the integrity and authenticity of the communication, and any cryptographic quantity remains secret.

#### 2.2.3.4 Rushing attack

Two colluded attackers use the tunnel procedure to form a wormhole. If a fast transmission path (e.g. a dedicated channel shared by attackers) exists between the two ends of the wormhole, the tunneled packets can propagate faster than those through a normal multi-hop route. This forms the rushing attack. The rushing attack can act as an effective denial of-service attack against all currently proposed on-demand WSN routing protocols, including protocols that were designed to be secure, such as ARAN and Ariadne [36].

#### 2.2.3.5 Selective Forwarding

In this type of attack, attacker refuses to forward packets or drop them and act as a black hole. A malicious node can selectively drop only certain packets. Especially effective if combined with an attack that gathers much traffic via the node. In sensor networks it is assumed that nodes faithfully forward received messages. But some compromised node might refuse to forward packets, however neighbors might start using another route. Even though the protocol is completely resistant to the sinkholes, wormholes, and the Sybil attack, a compromised node has a significant probability of including itself on a data flow to launch this type of attack if it is strategically located near the source or a base station [4].

#### 2.2.3.6 Misdirection

Changing or replaying the routing information can cause the misdirection attack. In this attack a malicious node, that is part of a route, can instead of dropping packets, quite simply send them on a different path where there does not exist a route to the destination [4,32]. Misdirection attack is also considered as routing layer attack.

#### 2.2.3.7 Hello Flood Attack

In Hello Flood Attack, attacker with high radio range sends more Hello packet to announce themselves to large number of nodes in the large network persuading themselves as neighbor [7].

#### 2.2.3.8 Blackhole Attack

The attacker node absorbs all the messages resulting loss of communication between nodes and therefore it can cause denial of service since it is dropping all the received packets. Alternately, the attacker node can monitor and analyze the traffic to find activity patterns of each node. Sometimes the black hole becomes the first step of a man-in-the-middle attack.

#### 2.2.3.9 Spoofing and Altering Routing Attack

In this attack, a malicious node may be able to create routing loops, wormholes, black holes, partition the network attract or repel network traffic, extend or shorten source routes, generate false error messages, increase end-to-end latency and etc., by spoofing, altering or replaying routing information [4].

#### 2.2.3.10 Routing table overflow attack

A malicious node advertises routes that go to non-existent nodes to the authorized nodes present in the network. It usually happens in proactive routing algorithms, which update routing information periodically. The attacker tries to create enough routes to prevent new routes from being created. The proactive routing algorithms are more vulnerable to table overflow attacks because proactive routing algorithms attempt to discover routing information before it is actually needed. An attacker can simply send excessive route advertisements to overflow the victim's routing table [8, 36].

#### 2.2.3.11 Routing cache poisoning attack

In route cache poisoning attacks, attackers take advantage of the promiscuous mode of routing table updating, where a node overhearing any packet may add the routing information contained in that packet header to its own route cache, even if that node is not on the path. Suppose a malicious node M wants to poison routes to node X. M could broadcast spoofed packets with source route to X via M itself; thus, neighboring nodes that overhear the packet may add the route to their route caches [8, 36].

#### 2.2.3.12 Simple Target Flooding

In this attack query packets are flooded inside the network to search for a certain target node. When the target node receives the query packet, it responds to the source node in order to inform the source node about its existence and to avoid further unnecessary flooding attempts from the source node. Even if the flooded packet reaches the target, packets flooded towards other directions continue.

#### 2.2.3.13 False Identity Target Flooding

Similar to simple target flooding, query packets are flooded inside the network to search for a certain target node, except the fact that the attacker deceives with wrong source ID [7].

#### 2.2.3.14 Fabrication Attack

Adversary may fabricate the routing messages to disorder the routing decisions. For instance, a malicious node could simply fabricate a route error message in AODV protocol, this will put all the upstream nodes in the network into a very embarrassment situation since all of them now believe that a certain number of destination are unable to reach. This may result in these upstream nodes to re-initiate a route request to those unreachable destinations so as to discover and build another possible route to them. This brings the energy consuming issue on the table again and significantly degrades the performance of the routing protocol [9].

#### 2.2.3.15 Tunneling Attack

In Ad hoc network, a node can be located adjacent to other nodes. A tunneling attack is referred to two or more malicious nodes in the network may collude and cooperate with each other to encapsulate and exchange routing messages between them by either using the existing data routes or potentially high power transceiver [9].

### 2.2.4 Transport Layer Attacks

#### 2.2.4.1 Flooding Attack

In this DoS attack, a malicious node may send continuous connection requests to a victim node effectively flooding the victim's network link [4]. Flooding attacks take place when adversary starts triggering multiple connection requests towards the target node. The adversary can be a legitimate node which has now been compromised otherwise an adversary may have higher capabilities, generating large number of legitimate packets and overwhelming the victim node [5].

The primary aim of flooding attacks is to cause exhaustion of resources on victim system. This process is analogous to TCP SYN attacks where, attacker sends many connection establishment requests, forcing the victim to store state of each connection request. The attack generates large volume of traffic that prevents legitimate user from accessing services. The main aim of this attack is either to block the node only or blocking link along with the node. As a result network performance decreases greatly [36].

#### 2.2.4.2 Desynchronisation

It can disrupt an existing connection between two end points. Adversary transmits forged packet with bogus sequence number or control flag to degrade or prevent the exchange of data [4].

### 2.2.5 Application Layer Attacks

#### 2.2.5.1 Message Corruption

In this case, any modification of the content of a message by an attacker that compromises its integrity is considered as message corruption attack. This action affects not only the data integrity which is needed to ensure the reliability of the data but also the ability to confirm that a message has not been tampered with, altered or changed [3].

#### 2.2.5.2 False Data Injection

In false data injection attacks, multiple compromised nodes collaboratively forge a fake report and inject the report into the network. This type of attacks is hard to defend with existing approaches, because they only verify a fixed number of message authentication codes (MACs) carried in the data report but the adversary can easily obtain enough compromised nodes from different geographical areas of the network to break their security [15].

#### 2.2.5.3 Masquerading

A bogus registration is an active attack which an attacker does a registration with a bogus re-of-address by masquerading itself as someone else. By advertising fraudulent beacons, an attacker might be able to attract a MN (mobile node) to register with the attacker as if MN has reached HA (home agent) or FA (foreign agent). Now, the attacker can capture sensitive personal or network data for the purpose of accessing network and may disrupt the proper functioning of network. It is difficult for an attacker to implement such type of attack because the attacker must have detailed information about the agent [10].

#### 2.2.5.4 Repudiation

Repudiation refers to the denial or attempted denial by a node involved in a communication of having participated in all or part of the communication. Example of repudiation attack is a commercial system in which a selfish person could deny conducting an operation on a credit card purchase or deny any on-line transaction Non-repudiation is one of the important requirements for a security protocol in any communication network [10].

#### 2.2.5.5 Key management Attack

Key management protocols deal with the key generation, storage, distribution, updating, revocation, and certificate service. Attackers can launch attacks to disclose the cryptographic key at the local host or during the key distribution procedure. The lack of a central trusted entity in WSN makes it more vulnerable to key management attacks [36].

## 2.3 Security Mechanisms for WSN

Security of a network is determined by the security over all layers. In the case of Wireless Sensor Networks, where the physical security of nodes is not guaranteed, arises the need of security measures that are resilient to physical tampering with nodes in the field operation.

According to [3, 42, 43] there are many security mechanisms that can be implemented in a WSN to ensure all the primary and secondary security goals that have been already mentioned in 2.1 section. Many of these security mechanisms focus on cryptography, key management, secure routing, secure data aggregation, secure location discovery, secure time synchronization, trust management systems and intrusion detection.

### 2.3.1Intrusion Detection Systems

An Intrusion Detection System plays an important role in WSN because it offers the ability to monitor a network, detect any unlawful action and alarm based on specific rules. Therefore it can eliminate most of the security attacks in a network in contrast to other security measures that ensure security only at some level. Despite that, security mechanisms in WSN are being studied for over twenty years, as implied in [32], intrusion detection systems are still in an immature level and only few of them are able to detect many separate attacks. Moreover, the limited resources of WSNs hamper the design and the implementation of such an efficient mechanism.

Intrusion Detection Systems are divided in categories according to the rules they apply to detect an intrusion. Anomaly-based IDSs are lightweight in nature; however they create more false alarms. Signature-based IDSs are suitable for relatively large-sized WSNs; however they have some overheads such as updating and inserting new signatures. Hybrid IDSs are a combination of both anomaly-based and signature-based approaches and are more energy consuming but more accurate in terms of attack detection with less number of false positives. Hybrid mechanisms usually contain two detection modules where one module is responsible of detecting well-known attacks using signatures and the other is responsible for detecting and learning normal and malicious patterns or monitor network behavior deviation from normal profile. Cross layer IDSs are usually not recommended for networks having resources limitations, as more energy and computation are required for exchanging multilayer parameters [76].

Additionally, it should be noted that it is still a challenge to prevent automatically and effectively an attack in wireless sensor networks even after its detection. After an attack detection a manual isolation or removal of the affected nodes takes place to limit the damage in the network.

Intrusion detection systems in wireless sensor networks are still a challenge for researchers and many of them are being proposed every year. In chapter 5 we perform a comparison analysis between recently proposed IDSs that include an experiment evaluation in their proposal.

# Chapter 3

# Experiment : Attacks in WSN

## 3.1 Simulation of WSN Attacks

There are many network simulators available. Some of the most popular network simulators are listed below:

- TOSSIM – Tiny OS Simulator
- ns-2 (C++)
- ns-3 (C++ & python for ns-3)
- OMNeT++ (C++)
- NetSim

For this experiment we have chosen the NS-2 because of its rich documentation, community and maturity in this specific area that we examine.

### 3.1.1 The NS-2 Simulator

The ns-2 is an open source event driven simulator used by the research community for research in networking. It has support for both wired and wireless networks and can simulate several network protocols such as TCP, UDP, multicast routing, etc. More recently, support has been added for simulation of large satellite and ad hoc wireless networks. The ns-2 simulation software was developed at the University of Berkeley. It is constantly under development by an active community of researchers.

The ns-2.35 is available as an all-in-one package that includes many modules. The standard ns-2 distribution runs on Linux. However, a package for running ns-2 on Cygwin (Linux Emulation for Windows) is available too. In the latter mode, ns-2 runs in the Windows environment on top of Cygwin.

The input to ns-2 is a Tcl script file. An NS2 simulation script (e.g., myfirst_ns.tcl) is referred to as a Tcl simulation script. Each script file corresponds to one specific experiment scenario and has the extension .tcl.

NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl).While the C++ defines the internal mechanism (i.e., a backend) of the simulation, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend). The C++ and the OTcl are linked together using TclCL [27].

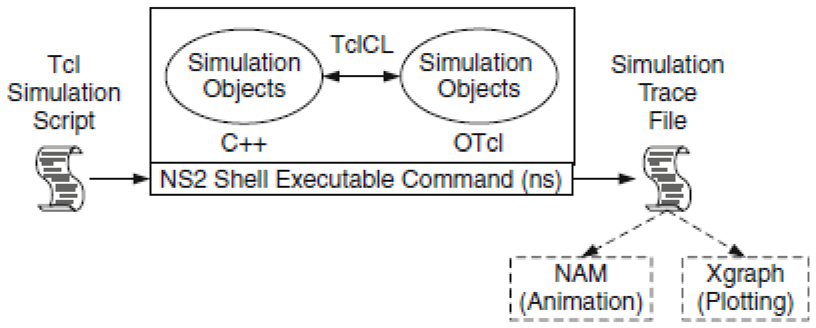

**Figure 2: Simulation flow**

The simulation can be classified into two categories: the time driven and the event driven simulation and is described in [27] as follows:

Time-driven simulation induces and executes events for every fixed time interval of Δ time units. In particular, it looks for events that may have occurred during this fixed interval. If found, such events would be executed as if they occurred at the end of this interval. After the execution, it advances the simulation clock by Δ time units and repeats the process. The simulation proceeds until the simulation time reaches a predefined termination time.

Time interval (Δ) is an important parameter of time-driven simulation. While a large interval can lead to loss of information, a small interval can cause unnecessary waste of computational effort.

An event-driven simulation does not proceed according to fixed time interval. Rather, it induces and executes events at any arbitrary time. Event-driven simulation has four important characteristics:

- Every event is stamped with its occurrence time and is stored in a so-called event list.
- Simulation proceeds by retrieving and removing an event with the smallest timestamp from the event list, executing it, and advancing the simulation clock to the timestamp associated with the retrieved event.

- At the execution, an event may induce one or more events. The induced events are stamped with the time when the event occurs and again are stored in the event list. The timestamp of the induced events must not be less than the simulation clock. This is to ensure that the simulation would never go backward in time.
- An event-driven simulation starts with a set of initial events in the event list. It runs until the list is empty or another stopping criterion is satisfied.

**NAM** [27] trace records simulation detail in a text file and uses the text file to play back the simulation using animation. NAM trace is activated by the command "$ns namtrace-all $file," where "ns" is the Simulator handle and "file" is a handle associated with the file (e.g., "out.nam" in the above example) which stores the NAM trace information. After obtaining a NAM trace file, the animation can be initiated directly at the command prompt through the "nam <filename.nam>" command.

**Xgraph** is a plotting program which can be used to create graphic representations of simulation results. First, there must be created output files in Tcl scripts, which can be used then as data sets for xgraph. In order to call xgraph to display the results using the command "xgraph <data-file>".

A **trace file** which contains many trace strings is created after a successful simulation run in NS2.
The format of a trace string is shown below:

| Type Identifier | Time | Source Node | Destination node | Packet name | Packet size | Flags | Flow ID | Source address | Destination address | Sequence number | Packet Unique ID |
|---|---|---|---|---|---|---|---|---|---|---|---|

12 fields of the trace string are as follows:
1. Type Identifier:
"+": a packet enque event
"-": a packet deque event
"r": a packet reception event
"d": a packet drop (e.g., sent to dropHead_) event
"c": a packet collision at the MAC level
**2. Time:** at which the packet tracing string is created.

**3-4. Source Node and Destination Node:** denote the IDs of the source and the destination nodes of the tracing object.

**5. Packet Name:** Name of the packet type

**6. Packet Size:** Size of the packet in bytes.

**7. Flags:** A 7-digit flag string

“-”: disable

1st = “E”: ECN (Explicit Congestion Notification) echo is enabled.

2nd = “P”: the priority in the IP header is enabled.

3rd : Not in use

4th = “A”: Congestion action

5th = “E”: Congestion has occurred.

6th = “F”: The TCP fast start is used.

7th = “N”: Explicit Congestion Notification (ECN) is on.

8. Flow ID

**9-10. Source Address and Destination Address:** the format of these two fields is “a.b”, where “a" is the address and "b" is the port.

11. Sequence Number

12. Packet Unique ID

NS-2 is a very powerful tool though does not provide yet any tool to analyze trace files. Consequently, the format of a trace string is a very useful knowledge because it helps to create scripts to analyze the simulation results.

## 3.2 Simulation Parameters

For any experiment, we have a set of control parameters which are specified by the user and a set of output parameters which we need to investigate upon. In the scenario based experiments, the set of input parameters are the parameters for the definition of the scenario and the specification of the traffic pattern.

### 3.2.1 Routing Protocol

There are many routing protocols that someone can implement in a Wireless Sensor Network. Below are mentioned some of the most common and most useful to implement. In the following chapter’s scenarios we chose to implement the AODV routing protocol.

#### 3.2.1.1 AODV

Ad hoc On-Demand Distance Vector Routing protocol (AODV) is an on demand routing protocol. In AODV, the communication involves main three procedures, i.e. path discovery, establishment and maintenance of the routing paths. AODV uses 3 types of control messages to run the algorithm, i.e. Route Request (RREQ), Route Reply (RREP) and Route Error (RERR) [25]. To find a route to the destination, the source node floods the network with RREQ packets. The RREQ packets create temporary route entries for the reverse path through every node it passes in the network. When it reaches the destination a RREP is sent back through the same path the RREQ was transmitted. Every node maintains a route table entry which updates the route expiry time.

#### 3.2.1.2 DSR

Dynamic Source Routing (DSR) is a routing protocol for wireless mesh networks. It is similar to AODV in that it forms a route on-demand when a transmitting computer requests one. However, it uses source routing instead of relying on the routing table at each intermediate device. Dynamic source routing protocol (DSR) is an on-demand protocol designed to restrict the bandwidth consumed by control packets in ad hoc wireless networks by eliminating the periodic table-update messages required in the table-driven approach. The major difference between this and the other on-demand routing protocols is that it is beacon-less and hence does not require periodic hello packet (beacon) transmissions, which are used by a node to inform its neighbors of its presence. The basic approach of this protocol (and all other on-demand routing protocols) during the route construction phase is to establish a route by flooding RouteRequest packets in the network. The destination node, on receiving a RouteRequest packet, responds by sending a RouteReply packet back to the source, which carries the route traversed by the RouteRequest packet received [26,30].

#### 3.2.1.3 DSDV

This protocol is based on classical Bellman-Ford routing algorithm designed for MANETS. Each node maintains a list of all destinations and number of hops to each destination. Each entry is marked with a sequence number. It uses full dump or incremental update to reduce network traffic generated by rout updates. The broadcast of route updates is delayed by settling time. The only improvement made here is avoidance of routing loops in a mobile network of routers. With this improvement, routing information can always be readily available, regardless of whether the source node requires the information or not. DSDV solve the problem of routing loops and count to infinity by associating each route entry with a sequence number indicating its freshness. In DSDV, a sequence number is linked to a destination node, and usually is originated by that node (the owner). The only case that a non-owner node updates a sequence number of a route is when it detects a link break on that route. An owner node always uses even-numbers as sequence numbers, and a non-owner node always uses odd-numbers. With the addition of sequence numbers, routes for the same destination are selected based on the following rules:

1. a route with a newer sequence number is preferred;
2. in the case that two routes have a same sequence number, the one with a better cost metric is preferred. [25]

#### 3.2.1.4 TORA

The Temporally-Ordered Routing Algorithm (TORA) is an algorithm for routing data across Wireless Mesh Networks or Mobile ad hoc networks. The TORA attempts to achieve a high degree of scalability using a "flat", non-hierarchical routing algorithm. In its operation the algorithm attempts to suppress, to the greatest extent possible, the generation of far-reaching control message propagation. In order to achieve this, the TORA does not use a shortest path solution, an approach which is unusual for routing algorithms of this type. TORA builds and maintains a Directed Acyclic Graph (DAG) rooted at a destination. No two nodes may have the same height [26,30].

Information may flow from nodes with higher heights to nodes with lower heights. Information can therefore be thought of as a fluid that may only flow downhill. By maintaining a set of totally-ordered heights at all times, TORA achieves loop-free multipath routing, as information cannot 'flow uphill' and so cross back on itself. The key design concept of TORA is localization of control messages to a very small set of nodes near the occurrence of a topological change. To accomplish this, nodes need to maintain the routing information about adjacent (one hop) nodes. The protocol performs three basic functions: Route creation, Route maintenance and Route erasure.

During the route creation and maintenance phases, nodes use a height metric to establish a directed acyclic graph (DAG) rooted at destination. Thereafter links are assigned based on the relative height metric of neighboring nodes. During the times of mobility the DAG is broken and the route maintenance unit comes into picture to reestablish a DAG routed at the destination. Timing is an important factor for TORA because the height metric is dependent on the logical time of the link failure. TORA's route erasure phase is essentially involving flooding a broadcast clear packet (CLR) throughout the network to erase invalid routes [26,30].

#### 3.2.1.5 GMR

Geographic multicast routing (GMR), is a multicast routing protocol for wireless sensor networks. It is a fully localized algorithm that efficiently delivers multicast data messages to multiple destinations. It does not require any type of flooding throughout the network. Each node propagating a multicast data message needs to select a subset of its neighbors as relay nodes towards destinations. GMR optimizes the cost over progress ratio where the cost is equal to the number of neighbors selected for relaying and the progress is the overall reduction of the remaining distances to destinations. Such neighbor selection achieves a good tradeoff between the bandwidth of the multicast tree and the effectiveness of the data distribution [21].

### 3.2.2 Topology

There are four basic types of wireless sensor data network topologies. The four WSN data network topologies are Peer to Peer, Star, Tree and Mesh. Each topology has its own pros and cons under the specific working environment constraints [20].

#### 3.2.2.1 Peer to Peer

Peer-to-Peer [20] networks allow each node to communicate directly with another node without needing to go through a centralized communications hub. Each Peer device is able to function as both a “client” and a “server” to the other nodes on the network.

#### 3.2.2.2 Star

Star [20] networks are connected to a centralized communications hub. Each node cannot communicate directly with one another; all communications must be routed through the centralized hub. Each node is then a “client” while the central hub is the “server”.

#### 3.2.2.3 Tree

Tree [20] networks use a central hub called a Root node as the main communications router. One level down from the Root node in the hierarchy is a Central hub. This lower level then forms a Star network. The Tree network can be considered a hybrid of both the Star and Peer to Peer networking topologies.

#### 3.2.2.4 Mesh

Mesh [20] networks allow data to “hop” from node to node, this allows the network to be self-healing. Each node is then able to communicate with each other as data is routed from node to node until it reaches the desired location. This type of network is one of the most complex and can cost a significant amount of money to deploy properly.

### 3.2.3 Performance Metrics

Quality of Service (QoS) is a demand in almost all forms of networks today. Moreover, measuring performance can indicate also whether a WSN is under attack or not. In order to proceed with the performance evaluation different metrics that would help in making comparisons must be selected. In the following scenarios the performance of the network in is studied by analyzing packet delivery ratio (PDR), packet drop ratio (PDrR), network throughput (NTh), end to end delay (EED) and energy consumption.

#### 3.2.3.1 Packet Delivery Ratio

Packet Delivery Ratio is defined as the ratio of the total number of data packets received by the destination node to the number of data packets sent by the source node as given in the following equation:

$$\boldsymbol{Packet\ Delivery\ Ratio\ (PDR)} = \frac{\sum \boldsymbol{of\ packets\ received\ by\ the\ destination\ node}}{\sum \boldsymbol{of\ packets\ sent\ by\ the\ source\ node}}$$

The greater value of packet delivery ratio means the better performance of the protocol [24].

#### 3.2.3.2 Packet Drop Ratio

Packet Delivery Drop is defined as the ratio of the total number of data packets dropped by the network to the number of data packets sent by the source node as given in the following equation:

$$\boldsymbol{Packet\ Drop\ Ratio\ (PDrR)} = \frac{\boldsymbol{\sum of\ packets\ dropped}}{\boldsymbol{\sum of\ packets\ sent\ by\ the\ source\ node}}$$

#### 3.2.3.3 Network Throughput

The network throughput (NTh) represents the average rate of successful message delivery over a communication channel within the threshold time. Throughput is measured using number of bits of packet received per unit time (normally, bps).

$$\boldsymbol{Network\ Throughput\ (NTh)} = \frac{\boldsymbol{\sum of\ packets\ generated\ by\ the\ source\ node}}{\boldsymbol{\sum\ of\ packets\ received\ by\ the\ destination\ node}}$$

As mentioned in [45] the major factors affecting throughput are: packet loss due to network congestion, available bandwidth, number of users in the network, data loss due to bit errors, improper queuing techniques used, usage of weighted fair queue or priority queue and slow start and multiple decrease techniques.

#### 3.2.3.4 Average End to End Delay

The average time taken by a data packet to arrive in the destination. An end to end delay includes all possible delay caused during route discovery, retransmission delay, queuing delay and relay time Only the data packets that successfully delivered to destinations are counted.

$$\boldsymbol{Delay\ (EED)} = \frac{\boldsymbol{\sum(arrive\ time - send\ time)}}{\boldsymbol{\sum Data\ packets\ received}}$$

#### 3.2.3.5 Energy consumption

Many attacks when applied cause too much energy consumption in order to exhaust the battery of a wireless sensor device. The energy consumption under an attack may differ from normal traffic energy consumption. The energy loss can be calculated by subtracting the amount of energy of a node after running a scenario from the node's initial amount of energy.

$$Energy\ loss\ =\ Initial\ amount\ of\ energy - Final\ amount\ of\ energy$$

## 3.2.4 Malicious Node in Routing Protocol

We need to modify the two files that are responsible for the AODV routing we are using. The header file aodv.h and the main source file aodv.cc

According to the installation steps described previously these files are located to the following paths:

```
/ns-allinone-2.35/ns-2.35/aodv/aodv.cc
/ns-allinone-2.35/ns-2.35/aodv/aodv.h
```

After the source code modifications to the aodv files (see Appendix) a number of nodes is set as malicious into the tcl scenario and won't forward a packet. In order to complete the procedure the NS-2 core files have to be updated with the modified aodv files. There must be produced a new aodv.o object file that reflects the modified aodv source code. Execute the following commands to complete:

```
$ make clean
$ make
$ sudo make install
```

## 3.3 Simulation Scenarios

Four scenarios are presented in this section. A normal traffic scenario is simulated in order to have the network measured in normal state. The other four scenarios represent four different attacks: the Blackhole attack, the Rushing attack, the Flooding attack and the Selective Forwarding attack.

In this work, we chose to examine Denial of Service attacks. Denial-of-Service attacks are recognized as one of the most serious threats due to the resources constrained property in WSN. Most of the WSN's routing protocols are vulnerable to such attacks. The Denial of Service attack is considered particularly as it targets the energy efficient protocols that are unique to wireless sensor networks [40].

We chose to analyze four popular Denial of Service attacks which all affect the routing and for that reason such attacks are also known as routing attacks. We selected four particular attacks because we noticed that almost all papers and research interest focus on them and therefore we concluded that they were the most valuable attacks to implement. However, the experiment becomes more interesting because the selected four attacks follow quite different methods to cause a denial of service: Blackhole, Flooding, Rushing and Selective Forwarding attack. The first one reduces the traffic by absorbing the packets while the second one increase the traffic by sending continuous connection requests to a victim node. The Rushing attack works in another manner too. It alters the routing information in order to form its attack through the legitimate nodes. Selective forwarding attack forms a denial of service attack in a more sophisticate way in order to stay longer undetected in network. The discrepancies between their executions make the detection of all DoS attacks using one methodology difficult.

Through this experiment we will be able to see if there is any common point in results between all the attacks. The simulation results show the impact of DoS attacks on performance of WSN.

### 3.3.1 Normal Traffic Scenario

Resuming the previous chapter, we create a Simulation environment consisting of 25 wireless mobile nodes which are placed uniformly and forming a Mobile Ad-hoc Network, moving about over a 1200 × 600 meters area for 120 seconds of simulated time. We have used standard two-ray ground propagation model, the IEEE 802.11 MAC, and Omni-directional antenna model of NS2. All values are formed in a table shown below:

| **Method** | **Value** |
|---|---|
| Channel type | Channel/Wireless channel |
| Radio-propagation model | Propagation/Two ray round |
| Network interface type | Phy/wirelessPhy |
| Mac Layer Protocol | Mac/802_11 |
| Routing Protocol | AODV |
| Traffic Model | CBR |
| Number of Sensor Nodes | 25 |
| Simulation Time (s) | 120s |
| Simulation Area (mxm) | 1200x600 |
| Transmission Rate | 0.1Mb |
| Node Initial Energy | 1000 J |

As source nodes are defined the nodes with node id 8, 16, 20, 21 and 22. As destination node is defined the node with node id 18. The total number of packets generated by all source nodes is 1242 packets, where the source node 8 generates 238 packets and every other source nodes generate 251 packets each. Also, we have to stress that through all scenarios the number of packets generated, source nodes, destination nodes and malicious nodes will remain the same.

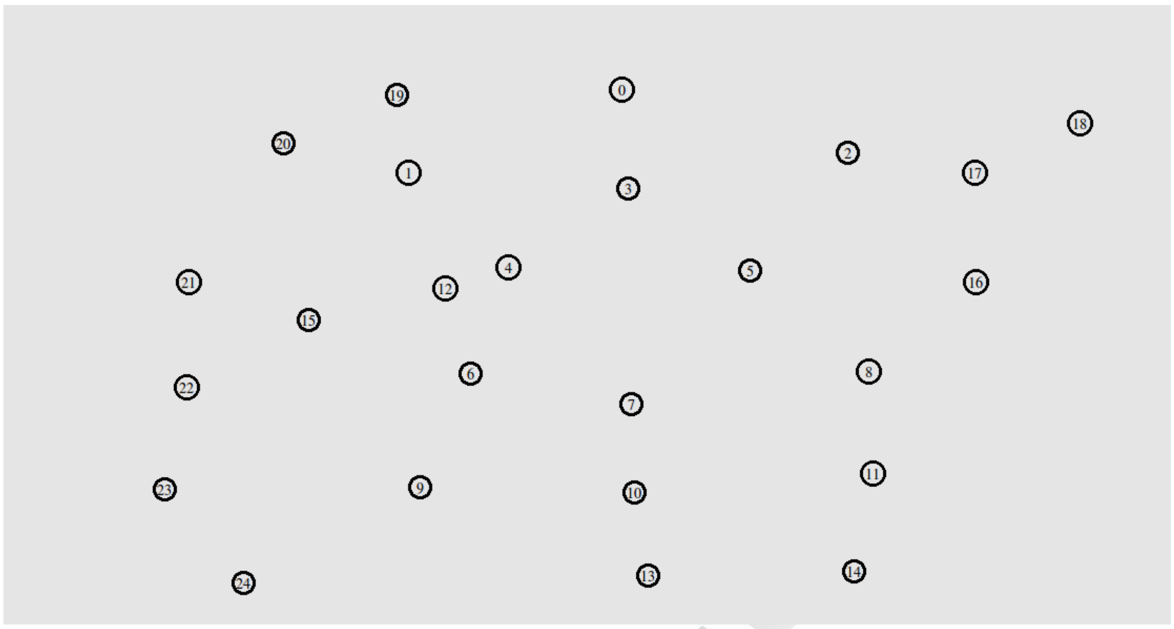

**Figure 3: Capture of our scenario simulation in NAM**

## 3.3.2 Blackhole Attack Scenario

In blackhole attack the malicious node sends a forged RREP packet to a source node that initiates the route discovery in order to pretend to be a destination node itself or a node of immediate neighbor the destination. Source node will forward all of its data packets to the malicious node; which were intended for the destination [33].

In order to apply this attack, we have to modify file aodv.cc, aodv.h inside the routing protocol. Blackhole always say that have the route to be a sink. So, the pseudocode for the modified routing protocol in C++ would be like:

```
else if ((rt && blackhole == 1)) {
    assert(rq> rq_dst == rt> rt_dst);
    sendReverse(rq> rq_src);
    rt> pc_insert(rt0> rt_nexthop);
    rt0> pc_insert(rt> rt_nexthop);
    Packet::free(p);}
```

The blackhole attack scenario follows the configuration of the normal traffic scenario. It consists of 25 nodes in which nodes with id 7, 8 and 10 are blackhole nodes and all the other nodes are non-malicious.

### 3.3.3 Rushing Attack Scenario

Rushing attack exploits this duplicate suppression mechanism by quickly forwarding route discovery packet in order to gain access to the forwarding group. When a node send a route request packet (RR packet) to another node in the wireless network, if there an attacker present then he will accept the RR packet and send to his neighbor with high transmission speed as compared to other nodes, which are present in the wireless network. Because of this high transmission speed, packet forwarded by the attacker will first reach to the destination node. Destination node will accept this RR packet and discard other RR packets which are reached later. Receiver found this route as a valid route and use for further communication. This way attacker will successfully gain access in the communication between sender and receiver [41].

On-demand routing protocols are more vulnerable to this attack, because whenever source node floods the route request packet in the network, an adversary node receives the route request packet and sends without any hop_count update and delay into the network. Rushing attack can be taken place at source side or destination side or at the middle. Also, this attack is more effective when attacker is near to source or destination node.

The following conditions the rushing attacker is not included in active route:

- When source and destination nodes have direct communication link
- When source and destination nodes have better route than rushing attackers route

Rushing attacks mainly classified into two types: i) Rushing attack followed by jellyfish attack and ii) Rushing attack followed by byzantine attack because it disturbs the data forwarding phase by either jellyfish or byzantine attack.

The rushing attack scenario follows the configuration of the normal traffic scenario as well. It consists of 25 nodes in which nodes with id 7, 8 and 10 are rushing attack nodes and all the other nodes are non-malicious.

### 3.3.4 Flooding Attack Scenario

As mentioned two sections before, a malicious node may send continuous connection requests to a victim node effectively flooding the victim’s network link.

The flooding attack can be launched by both external attacker and inside attacker. In case of external attacker we can use any of the authentication mechanisms and restrict the external attacker from entering the network and prevent the attack. In case of inside attack it is difficult to detect and the attack intensity is also more. The inside attacker behaves like normal node and send out the genuine route request but the only difference is that it send more amount of route request [39].

In this work, flooding attack is simulated in ns2 by using the timer based approach in AODV routing protocol. The malicious nodes with id 7, 8 and 10 will create more RREQs to a node, which is even doesn't exist in the network topology. In this scenario, the malicious nodes will create a RREQ to node 99 (which doesn't exist in the network) for every 0.09 seconds. This is how malicious node, start to flood the request in the network. The purpose of this attack is to consume the network bandwidth and to exhaust the network resources all the time.

### 3.3.5 Selective Forwarding Attack Scenario

In this scenario we implement an attack in which a malicious node can accomplish to form a type of blackhole attack but selectively. In this case the malicious node rather than attempting to drop all packets that come in it drop some packets selectively. This action can be done in many ways e.g. by dropping packets for a particular network destination, at a certain time of the day, a packet every *n* packets or every *t* seconds, or a randomly selected portion of the packets. This is rather called a selective forwarding attack or grayhole attack and it is often harder to detect because some traffic still flows across the network. This attack makes it more difficult to be discovered quickly through common networking tools such as traceroute because it adds a little complexity by changing its malicious nodes status from blackhole nodes to normal nodes. In the case of blackhole attack when other nodes notice that a compromised node is dropping all traffic, they begin to remove from their forwarding tables the malicious node but in this attack this method cannot be applied because malicious node becomes more undetectable.

In this scenario, the selective forwarding attack is implement by “flipping a coin” to decide if the node will behave as a normal node or as a drop packet node. We use this implementation of selective forwarding attack to verge on an average statistic of this type of attack. Same configuration as in normal traffic scenario applies and there are used 25 nodes. Nodes with id 7, 8 and 10 are the malicious and all the other nodes are non-malicious.

# Chapter 4

# Experiment Results

The trace file generated is the only output data from NS-2 that contains all the traffic flows' data and can be used to perform an analysis. In this experiment, NS2 Visual Trace Analyzer and a few awk scripts have been used in order to extract in a human readable format the statistics by the trace files. Also, a spreadsheet's program has been used to create visual representations of the resulted comparison tables.

Both the tcl file and the tracefile generated from the ns command execution for each scenario are imported into the NS2 Visual Trace Analyzer. The tool provides us the information for the average end to end delay, the total number of packets generated by all sources, the number of packets delivered and therefore the packets dropped. Moreover we used an awk script that calculates the Throughput of the network in each scenario. The calculation of the energy consumption was done manually through the trace files records. Comparative Analysis
In this subsection a comparative analysis is performed through the visualization of the data extracted from the trace files before.

### 4.1.1 Packet Delivery Ratio

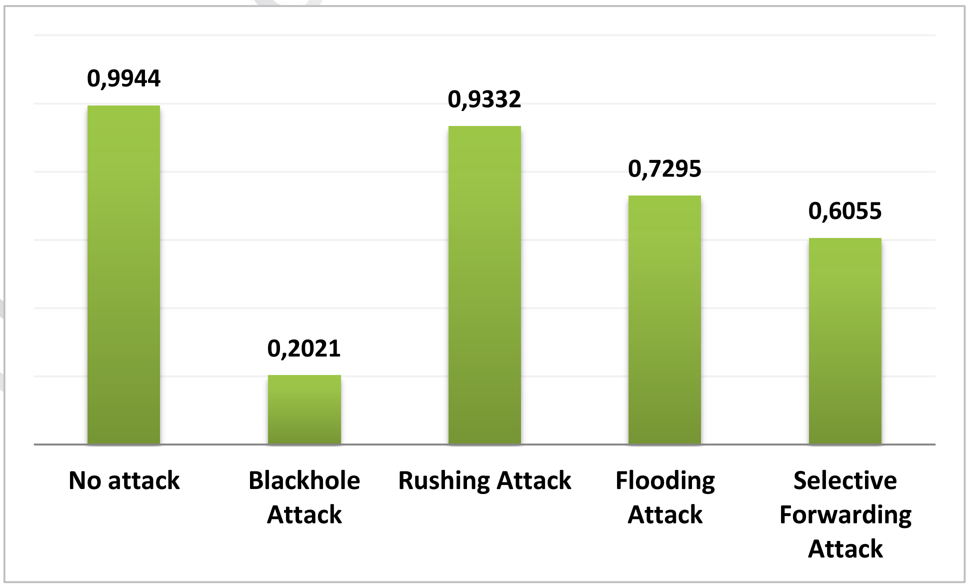

**Figure 4: Packet Delivery Ratio**

Figure 4 depicts the Packet Delivery Ratio for all simulated attack scenarios. The simulated attacks are represented in the x axis and the packet delivery ratio with range 0-1 in the y axis. Under normal traffic the packet delivery ratio is very high which means a good behavior of the network and a high quality of service. On the other hand, blackhole attack's scenario seems to have significant influence in Packet Delivery Ratio with ratio 0,2021 followed by selective forwarding attack which has a PDR of 0.6055 and then by flooding attack which has a PDR of 0.7295. Rushing attack affects almost 6% the PDR.

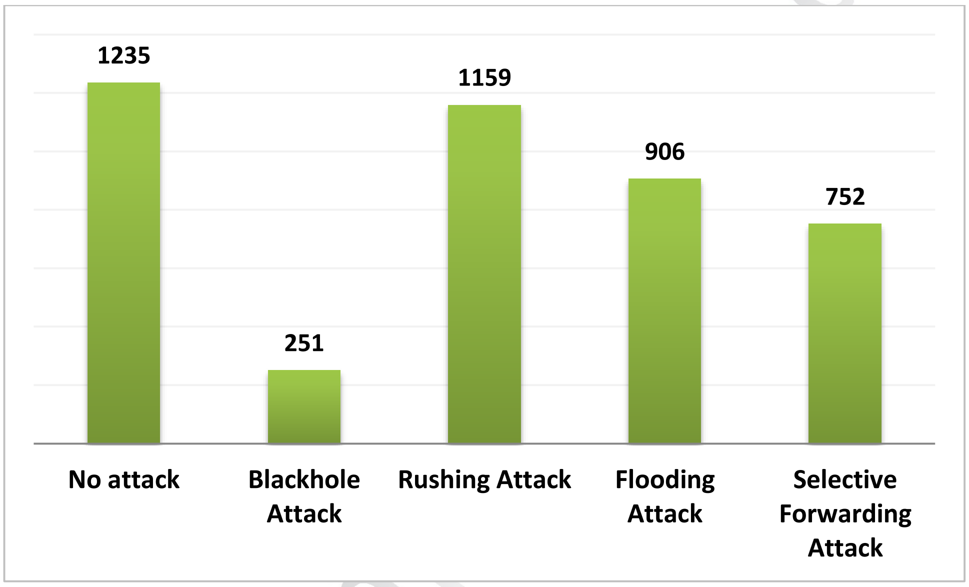

**Figure 5: Packet Delivery**

Figure 5 depicts the Packet Delivery for all simulated attack scenarios. It is the same graph as in Figure 4 with the only difference that here the simulated attacks are represented in the x axis and the number of packets delivered in the y axis. At this point, we should remind that the total number of packets generated by the network in all simulations is 1242 packets.

### 4.1.2 Packet Drop Ratio

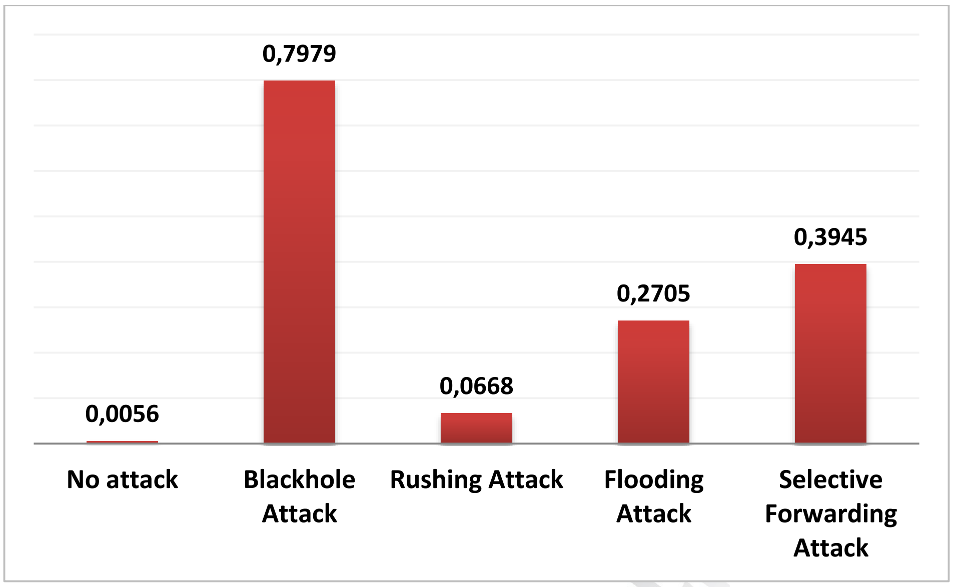

**Figure 6: Packet Drop Ratio**

Figure 6 depicts the Packet Drop Ratio for all simulated attack scenarios which is actually the complementary value of the previous chart. The simulated attacks are represented in the x axis and the packet drop ratio with range 0-1 in the y axis. In this chart we observe that the packet drop ratio is very low in normal traffic while the same network under blackhole attack faces a very high Packet Drop Ratio. Rushing attack introduces a small number of packet drops and flooding attack causes around 27% of packets to get dropped. Selective forwarding attack states its presence by introducing the second highest packet drop ratio after blackhole attack.

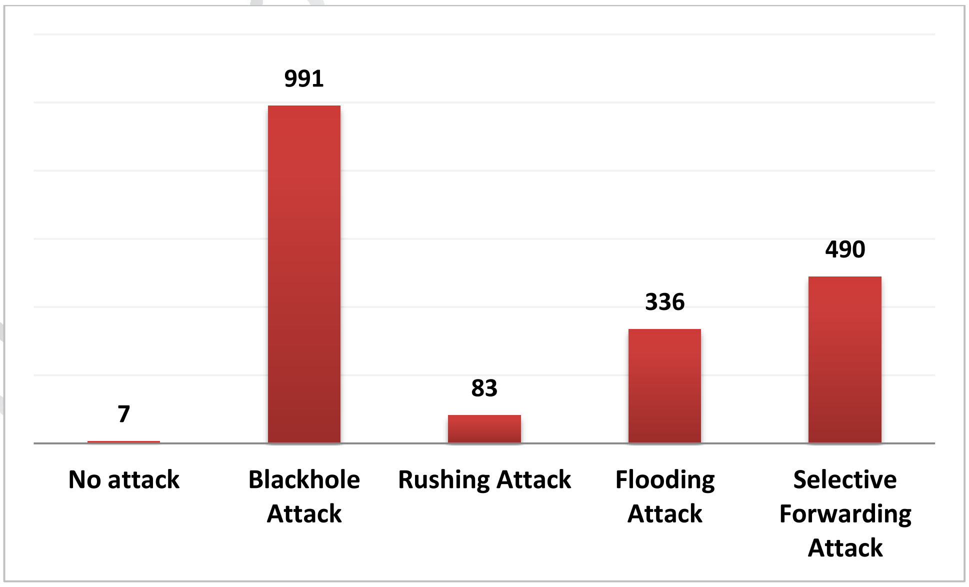

**Figure 7: Packet Loss**

Figure 7 depicts the Packet Loss for all simulated attack scenarios. It is the same graph as in Figure 6 with the only difference that here the simulated attacks are represented in the x axis and the number of packets delivered in the y axis. We should remind that the total number of packets generated by the network in all simulations is 1242 packets.

### 4.1.3 Throughput

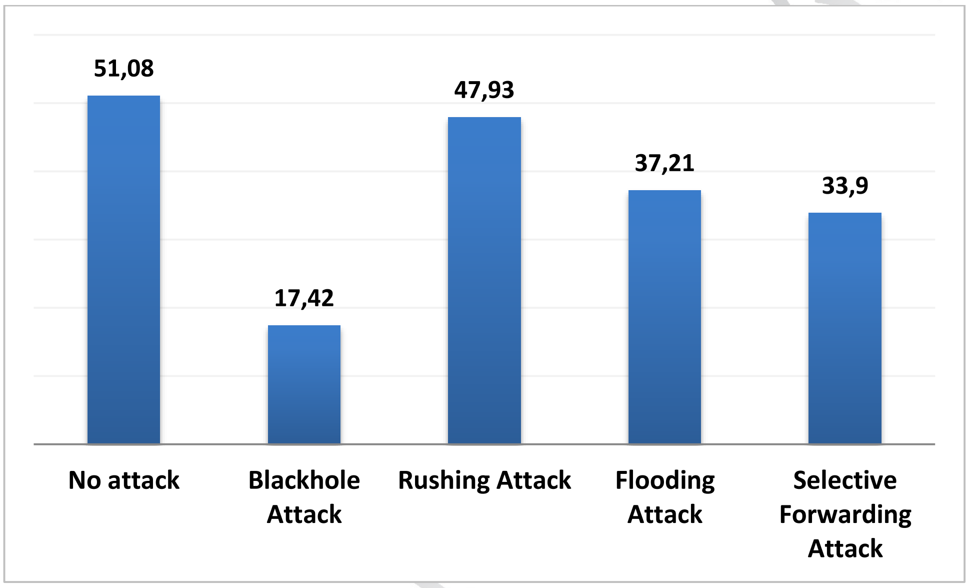

**Figure 8: Average Throughput in Kbps**

Figure 8 depicts the calculation of the Average Throughput for all simulated attack scenarios. The simulated attacks are represented in the x axis and the Kb per second are presented in the y axis. Once again, rushing attack presents a throughput close to the network's normal state throughput. Blackhole, Selective Forwarding and Flooding attacks decrease the throughput enough to make their presence obvious; with blackhole affecting the throughput much more than the flooding.

### 4.1.4 Average End to End Delay

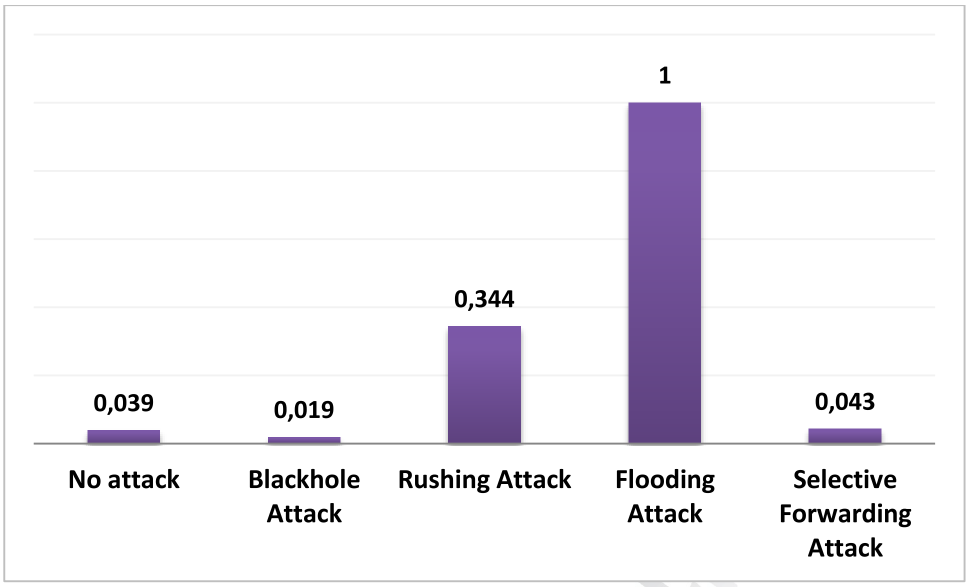

**Figure 9: Average End to End Delay**

Figure 9 depicts the average end to end delay (in seconds) of all packets after simulation. The attack that introduces the highest delay is flooding attack. However, the blackhole attack seems to improve the time of end-to-end delay because of its nature to drop all packets that pass by the malicious nodes. Rushing attack also inserts a delay in packet routing but not to the high level of the flooding attack.

### 4.1.5 Energy Consumption

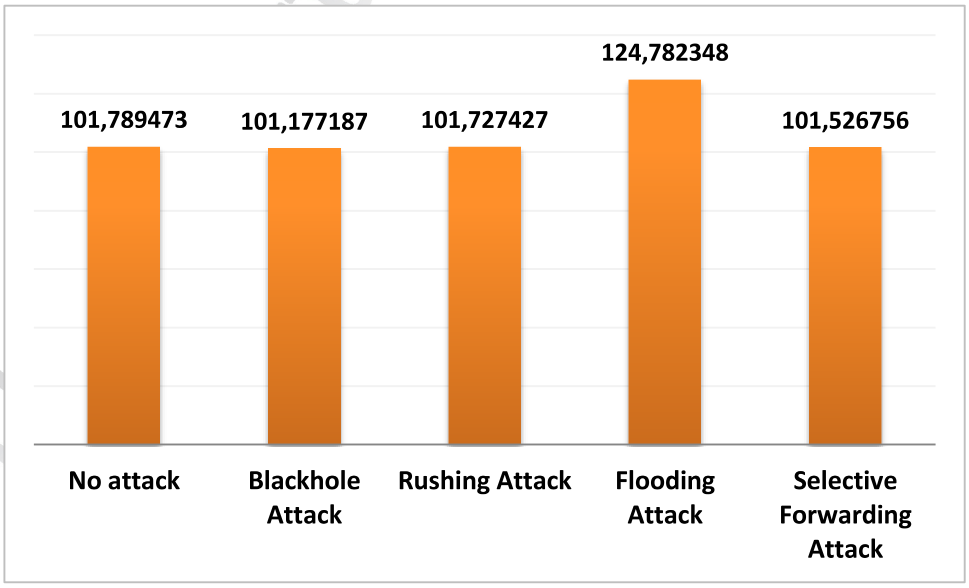

**Figure 10: Average energy consumption in Joules**

Figure 10 depicts the average energy consumption of all nodes after simulation. The initial energy of every node is 1000 J and for every scenario we measured the final energy in each node and we subtracted it by initial energy. In this way we calculated the average energy loss. In the figure above it is obvious that flooding attack consumes the most energy of all attacks and around 25% more energy than in normal circumstances. Rushing attack consumes almost the same energy as the network does under normal traffic which means that measuring energy to detect a possible attack in this kind of attack would probably fail. The average energy consumption in blackhole attack as well as in selective forwarding attack is lower than the average consumption in normal state. This is result is logically explained considering that the packets absorbed by the malicious nodes do not keep their route towards the destination node. An energy saving is resulted when packet is not following the next hops in route and the average energy consumption is in the end lower than it is normally.

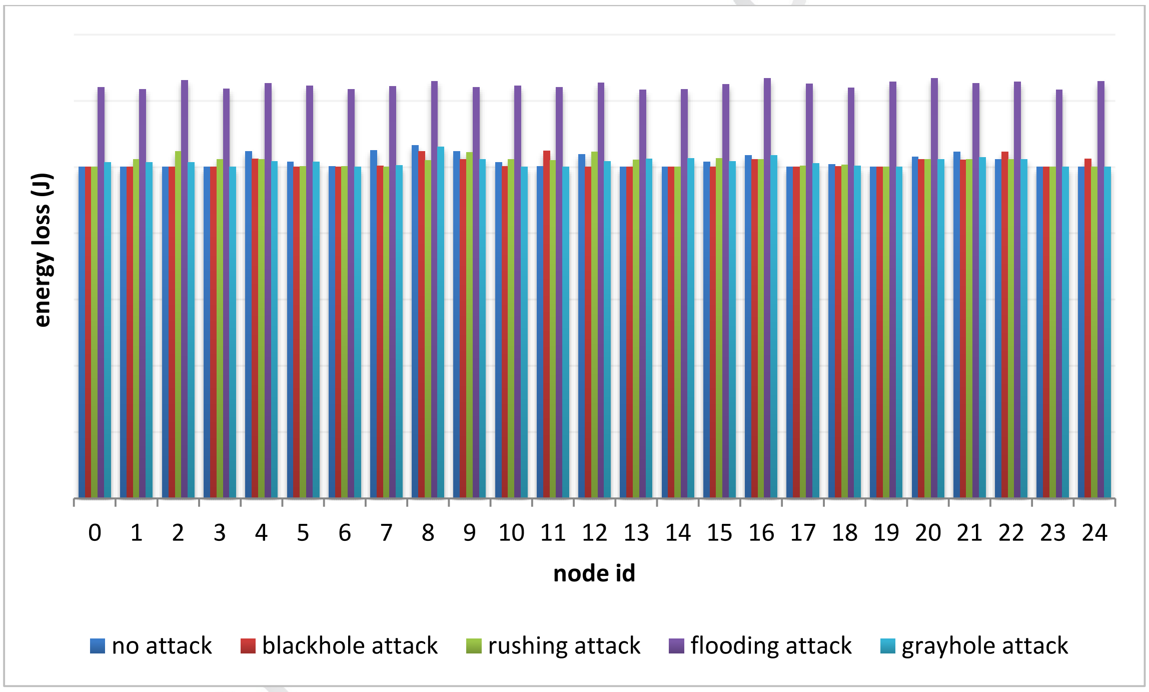

**Figure 11: Energy consumption of each node for all scenarios in Joule**

Figure 11 depicts the energy consumption of each node after simulation. The scope of this measurement is to observe if particular nodes appear to consume energy in contrast with other nodes. Moreover, we can test if the energy consumption of particular nodes such as source nodes, malicious nodes or the destination node is lower or higher than in normal traffic in order to find a pattern that could indicate the malicious nodes or the attack itself.

# Chapter 5

# IDS Effectiveness Evaluations

Researchers have suggested so far many approaches to detect harmful actions in wireless sensor networks but until today there is no IDS that is able to detect a very high percentage of all attacks. Moreover, many IDSs even though they are efficient in detecting attacks it is more likely to introduce overhead in communication, in computation or in energy consumption. Most of the times a proposed IDS is evaluated only against one or two metrics according to table 2 in [32].

Summarizing the two previous chapters, we conducted an experiment and for the performance evaluation we measured the packet delivery and packet loss ratio, the average end-to-end delay, the throughput and the energy consumption of a wireless sensor network in normal state and under four different denial-of-service attacks. In this section, having already analyzed our findings, we will match which of the previously used metrics are also used to evaluate the effectiveness of the most recent IDSs in wireless sensor networks. Packet Delivery Ratio and Packet Loss Ratio are complementary and for this comparative analysis there is no meaning to refer to both individually.

In [32] a complete comparative analysis is performed between 45 IDS mechanisms which are evaluated according to specific performance metrics. The representative sample of the evaluated IDS presents that in total the energy consumption has been evaluated by five existing IDSs, the packet drop ratio has been evaluated by three IDSs, the throughput has been evaluated by four IDSs and finally the average end to end delay has been evaluated by 4 existing IDSs. In general, only few of the proposed IDSs have been evaluated towards to the metrics related to the network performance.

In this work we update this table regarding the current status of proposed IDS mechanisms but we limit the table to the metrics we used in our experimental analysis. In this section 23 new IDSs that have been proposed and evaluated since mid-2012 are aggregated in the following table.

| | Energy consumption | Packet Delivery rate/Packet Loss rate | Throughput | EndToEnd Delay |
|---|---|---|---|---|
| Congestion aware IDS [46] | x | x | | x |
| Malicious and Malfunctioning [47] | | | | |
| Multivariate detection [48] | | x | | |
| Partially distributed IDS [49] | x | | | |
| Rule based IDS [50] | x | | x | |
| IAIDS [51] | | | | |
| OWIDS [52] | x | | | |
| BIAD [53] | | | | |
| FuzzyQ learning [54] | x | | | |
| Decision tree with wrapper [55] | | | | |
| Distributed fuzzy clustering [56] | | | x | |
| Distributed detection of mobile malicious nodes [57] | | | | |
| EID through Decision Trees [58] | | | | |
| Gaussian vs Uniform [59] | | | | |
| Hierarchical Node replication [60] | x | | x | |
| IDS for Heterogeneous & Homogeneous [61] | x | | | |
| Dual Threshold IDS [62] | | | | |
| MoteSec-Aware [63] | x | | | |
| Selective Forwarding IDS [64] | x | x | | |
| Sequential anomaly detection[65] | | x | | |
| Intrusion Detection Policy for WSN [66] | | x | x | |
| Subjective Logic-Based Anomaly Detection [67] | | | | |
| Watchdog LEACH [68] | x | | | |

In [46] the IDS selection method is evaluated towards energy consumption and it is claimed that it enhances the network's lifetime and reduces the total energy consumption.

In [47] malicious node detection rate (MDR), misdetection rate (MR), false alarm rate (FAR), and event detection accuracy (EDA), are defined to show the effectiveness of the proposed IDS.

Successful sending rate, Forwarding Rate, False Detection Rate and Node Density Influence on False Detection Rate are the metrics that are evaluated in the proposed IDS in [48].

In [49], except Memory Use and Energy consumption metrics, the Detection Rate is also used as a metric which is noticed to be increased when increasing average hop distance in the proposed scheme.

The evaluation metrics in [50] are Detection Rate, False Positives, Energy consumption and Reduce number of packets transmitted which affects throughput.

In [51] the proposed methodology is evaluated in terms of Detection rate and False alarm rate.

In [52], Energy consumption, Transmission accuracy, Attack detection rate, False positive rate and Accuracy rate are measuring the efficiency and the effectiveness of the IDS.

In [53] Detection rate, False alarm rate are used as metrics for the IDS evaluation.

Analysis of game-based FQL IDPS [54] is conducted in terms of detection accuracy, defense rate, number of live nodes and energy consumption.

In [55] the following metrics are used to evaluate IDS: Detection rate (Ratio between number of anomaly correctly classified and total number of anomaly). Error rate (Ratio between number of anomaly incorrectly classified and total number of anomaly). True positive (Classifying normal class as normal class). True negative (Classifying anomaly class as anomaly class). False positive (Classifying normal class as an anomaly class). False negative (Classifying anomaly class as a normal class).

In [56] the False Positive Rate and the False Negative Rate were calculated based on the observed number of False Positives and False Negatives for each of the three hierarchical stages. Also, the observed instances of True Positives and True Negatives are also determined at each stage. Then, the Sensitivity and Specificity values are calculated as follows to be the main evaluation metrics for data classification accuracy. Additionally, a communication overhead analysis is performed and temperature levels are measured.

In [57] the evaluation of the proposed scheme is done in terms of its detection capability, accuracy and speed.

Detection capabilities in [58] were measured through False Positive Rate (FPR), False Negative Rate (FNR), True Negative Rate (TNR), True Positive Rate (TPR), and Accuracy (ACC).

In order to evaluate the performance of intrusion detection in [59] they use the following two metrics: a) Intrusion distance which is the distance traveled by the the intruder before it is detected by a WSN for the first time and b) Detection probability which is defined as the probability that an intruder is detected within the maximal allowable intrusion distance, specified by a WSN application.

In [60] communication overhead is measured by the influence of clusters' size on the average number of packets sent and received for each node. Moreover, node replication detection probability, detection rate and energy consumption are used to evaluate the Hierarchical node replication IDS.

In [61] it is formed an evaluation of the detection probability and energy consumption.

Six metrics are used in [62]; malicious node detection rate (MDR), misdetection rate (MR), false alarm rate (FAR), event detection accuracy (EDA), event region detection rate (ERDR) and boundary false alarm rate (BFAR), are used to show the effectiveness of our scheme.

MoteSec-Aware [63] is evaluated by the energy consumption and the large-scale sensor network operations of the scheme.

The metrics considered for comparison in [64] are Energy left and Packet delivery ratio in presence and absence of malicious nodes.

The performance of the detection scheme proposed in [65] is evaluated by the detection latency and the receiver operating characteristic (ROC) curve, which is a plot of the detection rate versus the false alarm rate at different thresholds. They also plot the detection latency curve, which is a plot of the detection latency versus false alarm rates.

The proposed IDS in [66] is measuring its effectiveness against communication overhead by counting the number of sends, receive, forward and retransmit for each node. Therefore the influence packet delivery ratio and throughput is evaluated.

In [67] the evaluation of the IDS focuses exclusively in its detection capability using as evaluation metrics the detection rate, false detection rate, non-detection rate.

In [68] only energy consumption is measured to evaluate the proposed scheme.

The previous table shows that according to the whole number of the IDSs 10 of them have been evaluated in terms of energy consumption, 5 in terms of packet delivery/drop ratio, 4 in terms of throughput and 1 in terms of average end to end delay.

Most IDS are evaluated taking into consideration metrics such as False Positive Rate (FPR), False Negative Rate (FNR), True Negative Rate (TNR), True Positive Rate (TPR) and Accuracy (ACC). Moreover, this results to the fact that the proposed IDSs do not affect the network performance in terms of communication overhead, energy consumption, packet delivery ratio, etc.

# Conclusions

In this paper, it is performed a study of blackhole, flooding, rushing attack and selective forwarding attack launched in AODV routing protocol. All scenarios are conducted with the use of ns-2 and the simulation study depicts the performance degradation in terms of parameters like network throughput, average end to end delay, packet delivery ratio, packet loss and energy consumption.

From the results it can be observed that the presence of blackhole attack is obvious through the packet delivery ratio, the network throughput and the packet loss and not by the end to end delay or the energy consumption. The presence of flooding attack can also be detected by monitoring the packet delivery ratio, the average delay, the network throughput, the packet loss or the energy consumption. Rushing attack seems to be the most unlike attack to detect by simply monitoring the performance of the network. The results show that it presents values close to the normal traffic's statistics in almost all measurements except the average end to end delay and maybe the network throughput. Selective forwarding adds a little complexity by changing its malicious nodes status from blackhole nodes to normal nodes and proves that is more difficult to be discovered as quickly as blackhole. Although, it presence can be detected through the packet delivery ratio, the network throughput and the packet loss.

Although the attacks evaluated in experiment belong to the same group they do not affect the network in the same way. Furthermore, it is worth noting that these attacks are detected by the most of the IDSs.

Additionally, we used specific metrics to evaluate certain attacks and measure the effectiveness of some recently proposed IDSs in literature. Regarding our findings we conclude that network performance is affected both by routing attacks and the majority of the IDSs. The above conclusion makes inefficient the detection of such attacks by utilizing only certain network performance thresholds.

# References

[1] Shahriar Mohammadi, Reza Ebrahimi Atani, Hossein Jadidoleslamy, “A Comparison of Link Layer Attacks on Wireless Sensor Networks”, Journal of Information Security, p.69-84, 2011

[2] Dargie, W. and Poellabauer, C., “Fundamentals of wireless sensor networks: theory and practice”, John Wiley and Sons, ISBN 978-0-470-99765-9, pp. 168–183, 191–192, 2010

[3] Dr. G. Padmavathi, Mrs. D. Shanmugapriya, “A Survey of Attacks, Security Mechanisms and Challenges in Wireless Sensor Networks”, 2009

[4] Muhammad R Ahmed, Xu Huang, and Dharmendra Sharma, “A Taxonomy of Internal Attacks in Wireless Sensor Network”, 2012

[5] Rishav Dubey, Vikram Jain, Rohit Singh Thakur, Siddharth Dutt Choubey, “Attacks in Wireless Sensor Networks”, 2012

[6] Ana Paula R. da Silva Marcelo, H.T. Martins, Bruno P.S. Rocha, Antonio A.F. Loureiro, Linnyer B. Ruiz, Hao Chi Wong, “Decentralized Intrusion Detection in Wireless Sensor Networks”, Dept of Computer Science Federal Univ of Minas Gerais Belo Horizonte, MG, Brazil, 2005

[7] Mohammad Saiful Islam Mamun, A.F.M. Sultanul Kabir, “Hierarchical Design Based Intrusion Detection System For Wireless Ad Hoc Sensor Network”, Department of Computer Science, Stamford University Bangladesh, 51, Siddeshwari, Department of Computer Science and Engineering, American International University Bangladesh, Dhaka, 2012

[8] Teodor-Grigore Lupu, “Main Types of Attacks in Wireless Sensor Networks”, Department of Computer and Software Engineering University “Politehnica” of Timisoara, Faculty of Automatics and Computers Vasile Parvan 2, 300223, Timisoara, Romania, 2009

[9] Celia Li, Zhuang Wang, and Cungang Yang, “Secure Routing for Wireless Mesh Networks”, Department of Computer Science and Engineering, York University, M3J 1P3, Toronto, Canada , Department of Electrical and Computer Engineering, Ryerson University, 2010

[10] Noor Mohd, Singh Annapurna, H.S. Bhadauria, “Taxonomy on Security Attacks on Self Configurable Networks”, World Applied Sciences Journal 31 (3): 390-398, ISSN 1818-4952, IDOSI Publications, 2014

[11] Chaudhari H.C. and Kadam L.U., “Wireless Sensor Networks: Security, Attacks and Challenges”, International Journal of Networking, Volume 1, Issue 1, pp-04-16, 2011

[12] Harsh Kupwade Patil, Stephen A. Szygenda, “Security for Wireless Sensor Networks using Identity-Based Cryptography”, CRC Press, USA, 2013

[13] Pirzada Gauhar Arfaat, Dr. A.H. Mir,”The Impact of Wormhole Attack on the Performance of Wireless Ad-Hoc Networks”, IJCST Volume 2, Issue 4, ISSN : 0976-8491 (Online) | ISSN : 2229-4333 (Print), 2011

[14] Damandeep Kaur, Parminder Singh, “Various OSI Layer Attacks and Countermeasure to Enhance the Performance of WSNs during Wormhole Attack”, ACEEE, Int. J. on Network Security , Vol. 5, No. 1, January 2014

[15] Jianxin Wanga, Zhixiong Liua, b, Shigeng Zhanga, Xi Zhangc, “Information Sciences, Defending collaborative false data injection attacks in wireless sensor networks”, Information Sciences: an International Journal, Volume 254, p. 39-53, ISSN: 0020-0255, Elsevier Science Inc. New York, NY , USA, 2014

[16] Om Shree, Francis J. Ogwu, “A Proposal for Mitigating Multiple Black-Hole Attack in Wireless Mesh Networks”, Wireless Sensor Network, 5, p. 76-83, 2013

[17] T. Camp, J. Boleng, and V. Davies, “A Survey of Mobility Models for Ad Hoc Network Research”, Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol. 2, no. 5, pp. 483-502, 2002

[18] Md. Monzur Morshed, Meftah Ur Rahman,Md. Habibur Rahman, Md. Rafiqul Islam, “Performance Comparison of TCP variants over AODV, DSDV, DSR, OLSR in NS-2”, Department of Computer Science, American International University-Bangladesh, SCICON & TigerHATS Research Team-Bangladesh, George Mason University-USA, 2012

[19] Networking Wireless Sensors, Bhaskar Krishnamachari, Cambridge, 2005

[20] Steven Kosmerchock, “Wireless Sensor Network Topologies”, Goodyear

[21] Sanchez, J.A., Dept. of Commun. & Inf. Eng., Murcia Univ., Ruiz, P.M., Jennifer Liu, Stojmenovic, I., “Bandwidth-Efficient Geographic Multicast Routing Protocol for Wireless Sensor Networks”, Sensors Journal, IEEE (Volume:7 , Issue: 5 ), p. 627 – 636, 2007

[22] Achint Gupta, Dr. Priyanka V J, Saurabh Upadhyay, “Analysis of Wormhole Attack in AODV based MANET Using OPNET Simulator”, International Journal of Computing, Communications and Networking, Volume 1, No.2, ISSN 2319-2720, September–October 2012

[23] Bipul Syam Purkayastha, Rajib Das, “Comparative Analysis of Routing Attacks in Ad Hoc Network”, Int. J. Advanced Networking and Applications, Volume: 03 Issue: 05 Pages: 1352-1357, 2012

[24] Narendra Kumar Agarwal, Vishal Shrivastava, “Simulation Results and Performance Evaluation of Routing Protocols in Mobile Ad-Hoc Networks”, International Journal of Emerging Science and Engineering (IJESE), ISSN:2319-6378, Volume-1, Issue-7, 2013

[25] Harris Simaremare, Riri Fitri Sari, “Performance Evaluation of AODV variants on DDOS, Blackhole and Malicious Attacks”, IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.6, 2011

[26] Bikash Rath, “Implementing And Comparing Dsr And Dsdv Routing Protocols For Mobile Ad Hoc Networking”, Department of Computer Science and Engineering National Institute of Technology, Rourkela, 2009

[27] T. Issaraiyakul and E. Hossain, “Introduction to Network Simulator NS2”, Springer, 2009

[28] Vandana C.P, Dr. A. Francis Saviour Devaraj, “Evaluation of Impact of Wormhole Attack on AODV”, Int. J. Advanced Networking and Applications, Volume: 04 Issue: 04 p. 1652-1656, ISSN : 0975-0290, 2013

[29] Mehul Revankar , “Attacks in Ad-Hoc Networks and Modeling in NS-2”, 2005

[30] Rakesh Kumar Jha , Suresh V. Limkar, Dr. Upena D. Dalal, “A Performance Comparison of Routing Protocols(DSR and TORA) for Security Issue In MANET(Mobile Ad Hoc Networks)”, IJCA Special Issue on “Mobile Ad-hoc Networks” MANETs, 2010

[31] Monika Roopak , Dr. Bvr Reddy¸ “Performance Analysis of Aodv Protocol under Black Hole Attack”, International Journal of Scientific & Engineering Research Volume 2, Issue 8, ISSN 2229-5518, August 2011

[32] Eleni Darra, Sokratis Katsikas, “Attack Detection Capabilities of Intrusion Detection Systems for Wireless Sensor Networks”, Department of Digital Systems, University of Piraeus, Greece, 2013

[33] Dr. Deepali Virmani, Manas Hemrajani, Shringarica Chandel, “Exponential Trust Based Mechanism to Detect Black Hole attack in Wireless Sensor Network”, Cornell University, USA, 2014

[34] Shafiullah Khan Al-Sakib Khan Pathan, “Wireless Networks and Security: Issues, Challenges and Research Trends”, ISBN:9783642361692, Springer, 2013

[35] Vikash Kumar, Anshu Jain, P N Barwal, “Wireless Sensor Networks: Security Issues, Challenges and Solutions”, International Journal of Information & Computation Technology, ISSN 0974-22 39 Volume 4, Number 8 (2014), pp.859-868, India, 2014

[36] Yang Xiao, Xuemin (Sherman) Shen, Ding-Zhu Du, “Wireless Network Security”, Springer, 2007

[37] Prateek Suraksha Bhushan, Abhishek Pandey & R.C.Tripathi ”A scheme for Prevention of Flooding Attack in Wireless sensor Network”,Vol 1 No. 2 June 2011

[38] Kalpana Sharma, M K Ghose, “Wireless Sensor Networks: An Overview on its Security Threats”, IJCA Special Issue on “Mobile Ad-hoc Networks” MANETs, India, 2010

[39] Bhuvaneshwari. K , Dr.A.Francis Saviour Devaraj, “Examination of Impact of Flooding attack on MANET and to accentuate on Performance Degradation”, Int. J. Advanced Networking and Applications Volume: 04 Issue: 04 Pages:1695-1699 (2013) ISSN : 0975-0290, India, 2013

[40] Doddapaneni.Krishna Chaitanya, Ghosh.Arindam, “Analysis of Denial of Service attacks on Wireless Sensor Networks Using Simulation”, 2013

[41] Satyam Shrivastava, “Rushing Attack and its Prevention Techniques”, International Journal of Application or Innovation in Engineering & Management, Volume 2, Issue 4, ISSN 2319-4847, 2013

[42] John Paul Walters, Zhengqiang Liang,Weisong Shi, and Vipin Chaudhary “Wireless Sensor Network Security: A Survey”, Security in Distributed, Grid, and Pervasive Computing, 2006

[43] Xiuli Ren, Haibin Yu, “Security Mechanisms for Wireless Sensor Networks”, JCSNS International Journal of Computer Science and Network Security, VOL.6 No.3, 2006

[44] A Survey on Security Mechanisms and Attacks in Wireless Sensor Networks”, nternational Journal of Engineering and Innovative Technology (IJEIT) Volume 2, Issue 3, September 2012

[45] P.N.Renjith and E.Baburaj, “Analysis on Ad Hoc Routing Protocols in Wireless Sensor Networks”, International Journal of Ad hoc, Sensor & Ubiquitous Computing (IJASUC) Vol.3, No.6, December 2012

[46] Jaeun Choi, Gisung Kim, Sehun Kim, “A Congestion-Aware IDS Node SelectionMethod forWireless Sensor Networks” Hindawi Publishing Corporation, International Journal of Distributed Sensor Networks, Volume 2012, Article ID 582139, 6 pages, 2012

[47] Seo Hyun Oh, Chan O. Hong, Yoon-Hwa Choi, “A Malicious and Malfunctioning Node Detection Scheme for Wireless Sensor Networks”, Wireless Sensor Network, 4, p. 84-90, 2012

[48] Hongjun Dai, Huabo Liu, Zhiping Jia, and Tianzhou Chen. “A Multivariate Classification Algorithm for Malicious Node Detection in Large-Scale WSNs”, In Proceedings of the 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TRUSTCOM '12). IEEE Computer Society, Washington, DC, USA, 239-245, 2012

[49] Eung Jun Cho, Choong Seon Hong, Sungwon Lee and Seokhee Jeon, “A Partially Distributed Intrusion Detection System for Wireless Sensor Networks”, Sensors, 13, p. 15863-15879, ISSN 1424-8220, 2013

[50] K. Anand, S. Ganapathy, K. Kulothungan, P. Yogesh, A. Kannan, “A Rule Based Approach for Attribute Selection and Intrusion Detection in Wireless Sensor Networks”, Procedia Engineering, Volume 38, p. 1658-1664, ISSN 1877-7058, 2012

[51] Ranjit Panigrahi, Kalpana Sharma and M k Ghose, “An Integrated Approach of Intrusion Detection System (IAIDS) for Wireless Sensor Networks”, IJCA Proceedings on Computing Communication and Sensor Network 2013 CCSN 2013(2):5-8, December 2013

[52] Chia-Fen Hsieh, Rung-Ching Chen, and Yung-Fa Huang, “Applying an Ontology to a Patrol Intrusion Detection System for Wireless Sensor Networks”, International Journal of Distributed Sensor Networks, Volume 2014, Article ID 634748, 14 pages, 2014

[53] Rongrong Fu, Kangfeng Zheng, Tianliang Lu, Dongmei Zhang, Yixian Yang ,“Biologically Inspired Anomaly Detection for Hierarchical Wireless Sensor Networks”, Journal of Networks, Vol 7, No 8 (2012), 1214-1219, Aug 2012

[54] Shahaboddin Shamshirband, Ahmed Patel, Nor Badrul Anuar, Miss Laiha Mat Kiah, Ajith Abraham, “Cooperative game theoretic approach using fuzzy Q-learning for detecting and preventing intrusions in wireless sensor networks”, Engineering ApplicationsofArtificial Intelligence32(2014)228–241, 2014

[55] Siva S. Sivatha Sindhu, S. Geetha, A. Kannan, “Decision tree based light weight intrusion detection using a wrapper approach”, Expert Systems with Applications, p. 129–141, 2012

[56] Heshan Kumarage, Ibrahim Khalil, Zahir Tari, Albert Zomaya “Distributed anomaly detection for industrial wireless sensor networks based on fuzzy data modelling”, Journal of Parallel and Distributed Computing archive, Volume 73 Issue 6, p. 790-806, June 2013

[57] Jun-Won Ho, Matthew Wright, Sajal K. Das, “Distributed detection of mobile malicious node attacks in wireless sensor networks”, Ad Hoc Networks, Volume 10, Issue 3, p. 512-523, May 2012

[58] Alessia Garofalo, Cesario Di Sarno, Valerio Formicola, “Enhancing Intrusion Detection in Wireless Sensor Networks through Decision Trees”, Dependable Computing, Lecture Notes in Computer Science Volume 7869, p. 1-15, 2013

[59] Yun Wang, Weihuang Fu, Dharma P. Agrawal, "Gaussian versus Uniform Distribution for Intrusion Detection in Wireless Sensor Networks," IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 2, pp. 342-355, Feb. 2013

[60] Wassim Znaidi, Marine Minier, and Stéphane Ubéda, “Hierarchical Node Replication Attacks Detection in Wireless Sensor Networks,” International Journal of Distributed Sensor Networks, vol. 2013, Article ID 745069, 12 pages, 2013.

[61] Jasvinder Singh, Er. Vivek Thapar, “Intrusion Detection in Wireless Sensor Network”, International Journal of Computer Science and Communication Engineering, Volume 1 Issue 2, ISSN: 2319-7080, December 2012

[62] Sung Yul Lim, Yoon-Hwa Choi, "Malicious Node Detection Using a Dual Threshold in Wireless Sensor Networks" Journal of Sensor and Actuator Networks, ISSN 2224-2708, p. 70-84, 2013

[63] Yao-Tung Tsou, Chun-Shien Lu and Sy-Yen Kuo, "MoteSec-Aware: A Practical Secure Mechanism for Wireless Sensor Networks", IEEE Transactions On Wireless Communications, Vol. 12, No. 6, June 2013

[64] Bharti Bains, Rohit Vaid, "Selective Forwarding based Intrusion Detection System for Secure Wireless Sensor Network, International Journal of Computer Applications (0975 – 8887), Volume 77– No.13, September 2013

[65] S. Zheng & J. S. Baras, "Sequential Anomaly Detection in Wireless Sensor Networks and Effects of Long-Range Dependent Data", Sequential Analysis: Design Methods and Applications, 31:4, p. 458-480, 2012

[66] Jiang Xu, Jin Wang, Shengdong Xie, Wenliang Chen and Jeong-Uk Kim, "Study on Intrusion Detection Policy for Wireless Sensor Networks", International Journal of Security and its Applications, Vol.7, No.1, January 2013

[67] Jinhui Yuan, Hongwei Zhou, and Hong Chen, "Subjective Logic-Based Anomaly Detection Framework in Wireless Sensor Networks," International Journal of Distributed Sensor Networks, vol. 2012, Article ID 482191, 13 pages, 2012

[68] Mohammad Reza Rohbanian, Mohammad Rafi Kharazmi, Alireza Keshavarz-Haddad, Manije Keshtgary, "Watchdog-LEACH: A new method based on LEACH protocol to Secure Clustered Wireless Sensor Networks", ACSIJ Advances in Computer Science: an International Journal, Vol. 2, Issue 3, No. 4, ISSN : 2322-5157, July 2013

[69] Keshav Goyal, Nidhi Gupta, Keshawanand Singh, "A Survey on Intrusion Detection in Wireless Sensor Networks" ,International Journal of Scientific Research Engineering & Technology (IJSRET), Volume 2, Issue2, ISSN 2278–0882, p. 113-126, May 2013

[70] Kashyap Patel , Mrs. T. Manoranjitham, "Detection of Wormhole Attack In Wireless Sensor Network", International Journal of Engineering Research & Technology (IJERT), Vol. 2 Issue 5, ISSN: 2278-0181, May 2013

[71] Ruchi Bhatnagar and Udai Shankar, "The Proposal of Hybrid Intrusion Detection for Defence of Sync Flood Attack in Wireless Sensor Network",

International Journal of Computer Science & Engineering Survey (IJCSES) Vol.3, No.2, April 2012

[72] Nabil Ali Alrajeh and J. Lloret, “Intrusion Detection Systems Based on Artificial Intelligence Techniques in Wireless Sensor Networks”, Hindawi Publishing Corporation, International Journal of Distributed Sensor Networks, Volume 2013, Article ID 351047, 6 pages, 2013

[73] Murad A. Rassam, M.A. Maarof and Anazida Zainal, “A survey of Intrusion Detection Schemes in Wireless Sensor Networks”, American Journal of Applied Sciences 9 (10): 1636-1652, ISSN 1546-9239, 2012

[74] Helio Mendes Salmon, Claudio M. de Farias, Paula Loureiro, Luci Pirmez, Silvana Rossetto, Paulo Henrique de A. Rodrigues, Rodrigo Pirmez, Flávia C. Delicato, Luiz Fernando R. da Costa Carmo, “Intrusion Detection System for Wireless Sensor Networks Using Danger Theory Immune-Inspired Techniques”, International Journal of Wireless Information Networks, Volume 20, Issue 1, pp 39-66, March 2013

[75] Nabil Ali Alrajeh, S. Khan, and Bilal Shams, “Intrusion Detection Systems in Wireless Sensor Networks: A Review,” International Journal of Distributed Sensor Networks, vol. 2013, Article ID 167575, 7 pages, 2013

[76] Priyanka Shah, Dr. Atul Patel, “Incremental Intrusion Detection System for Wireless Sensor Networks”, International Journal of Emerging Trends & Technology in Computer Science, Volume 2, Issue 6, ISSN 2278-6856, 2013

# Appendix

## Normal traffic Code

### normaltraffic.tcl

```
#====================================
#     Simulation parameters setup
#====================================
set val(chan)   Channel/WirelessChannel   ;# channel type
set val(prop)   Propagation/TwoRayGround  ;# propagation model
set val(netif)  Phy/WirelessPhy           ;# network interface type
set val(mac)    Mac/802_11                ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue   ;# interface queue type
set val(ll)     LL                        ;# link layer type
set val(ant)    Antenna/OmniAntenna       ;# antenna model
set val(ifqlen) 50                        ;# max packet in ifq
set val(nn)     25                 ;# number of mobilenodes
set val(rp)     AODV               ;# routing protocol
set val(x)      1200               ;# X dimension of topography
set val(y)      600                ;# Y dimension of topography
set val(stop)   120               ;# time of simulation end (s)
set val(energymodel)    EnergyModel     ;
set val(radiomodel)     RadioModel      ;
set val(initialenergy)  1000            ;# Initial energy (J)

#====================================
#        Initialization
#====================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open normal.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open normal.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];# Create wireless channel
```

```
#====================================
#     Mobile node parameter setup
#====================================
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      OFF \
                -movementTrace ON \
                -energyModel   $val(energymodel) \
                -idlePower 1.0 \
                -rxPower 1.0 \
                -txPower 2.0 \
                -sleepPower 0.001 \
                -transitionPower 0.2 \
                -transitionTime 0.005 \
                -initialEnergy $val(initialenergy)


#====================================
#        Nodes Definition
#====================================
#Create 25 nodes
set n0 [$ns node]
$n0 set X_ 663
$n0 set Y_ 484
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 466
$n1 set Y_ 407
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 871
$n2 set Y_ 426
$n2 set Z_ 0.0
```

```
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 668
$n3 set Y_ 393
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 558
$n4 set Y_ 320
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 781
$n5 set Y_ 317
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 523
$n6 set Y_ 222
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 671
$n7 set Y_ 194
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20
set n8 [$ns node]
$n8 set X_ 891
$n8 set Y_ 224
$n8 set Z_ 0.0
$ns initial_node_pos $n8 20
set n9 [$ns node]
$n9 set X_ 476
$n9 set Y_ 117
$n9 set Z_ 0.0
$ns initial_node_pos $n9 20
set n10 [$ns node]
$n10 set X_ 674
$n10 set Y_ 112
$n10 set Z_ 0.0
$ns initial_node_pos $n10 20
set n11 [$ns node]
$n11 set X_ 895
$n11 set Y_ 130
$n11 set Z_ 0.0
```

```
$ns initial_node_pos $n11 20
set n12 [$ns node]
$n12 set X_ 500
$n12 set Y_ 300
$n12 set Z_ 0.0
$ns initial_node_pos $n12 20
set n13 [$ns node]
$n13 set X_ 687
$n13 set Y_ 36
$n13 set Z_ 0.0
$ns initial_node_pos $n13 20
set n14 [$ns node]
$n14 set X_ 877
$n14 set Y_ 39
$n14 set Z_ 0.0
$ns initial_node_pos $n14 20
set n15 [$ns node]
$n15 set X_ 373
$n15 set Y_ 271
$n15 set Z_ 0.0
$ns initial_node_pos $n15 20
set n16 [$ns node]
$n16 set X_ 990
$n16 set Y_ 306
$n16 set Z_ 0.0
$ns initial_node_pos $n16 20
set n17 [$ns node]
$n17 set X_ 989
$n17 set Y_ 407
$n17 set Z_ 0.0
$ns initial_node_pos $n17 20
set n18 [$ns node]
$n18 set X_ 1086
$n18 set Y_ 453
$n18 set Z_ 0.0
$ns initial_node_pos $n18 20
set n19 [$ns node]
$n19 set X_ 455
$n19 set Y_ 479
$n19 set Z_ 0.0
$ns initial_node_pos $n19 20
set n20 [$ns node]
$n20 set X_ 350
$n20 set Y_ 434
$n20 set Z_ 0.0
```

```
$ns initial_node_pos $n20 20
set n21 [$ns node]
$n21 set X_ 263
$n21 set Y_ 306
$n21 set Z_ 0.0
$ns initial_node_pos $n21 20
set n22 [$ns node]
$n22 set X_ 261
$n22 set Y_ 209
$n22 set Z_ 0.0
$ns initial_node_pos $n22 20
set n23 [$ns node]
$n23 set X_ 240
$n23 set Y_ 115
$n23 set Z_ 0.0
$ns initial_node_pos $n23 20
set n24 [$ns node]
$n24 set X_ 313
$n24 set Y_ 29
$n24 set Z_ 0.0
$ns initial_node_pos $n24 20


#====================================
#        Generate movement
#====================================
$ns at 0 " $n6 setdest 1086 453 40 "
$ns at 10 " $n18 setdest 877 39 40 "
$ns at 20 " $n18 setdest 500 117 40 "
$ns at 60 " $n18 setdest 400 100 40 "
$ns at 40 " $n6 setdest 400 500 40 "
$ns at 10 " $n15 setdest 650 470 40 "
$ns at 10 " $n5 setdest 550 220 40 "

#====================================
#        Agents Definition
#====================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n21 $udp0
set null1 [new Agent/Null]
$ns attach-agent $n18 $null1
$ns connect $udp0 $null1
$udp0 set packetSize_ 1500
```

```
#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 0.1Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 20.0 "$cbr0 stop"
#Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n20 $udp1
set null2 [new Agent/Null]
$ns attach-agent $n18 $null2
$ns connect $udp1 $null1
$udp1 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 0.1Mb
$cbr1 set random_ null
$ns at 20.0 "$cbr1 start"
$ns at 40.0 "$cbr1 stop"
#Setup a UDP connection
set udp3 [new Agent/UDP]
$ns attach-agent $n22 $udp3
set null3 [new Agent/Null]
$ns attach-agent $n18 $null3
$ns connect $udp3 $null1
$udp3 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp3
$cbr2 set packetSize_ 1000
$cbr2 set rate_ 0.1Mb
$cbr2 set random_ null
$ns at 40.0 "$cbr2 start"
$ns at 60.0 "$cbr2 stop"
set udp4 [new Agent/UDP]
$ns attach-agent $n8 $udp4
set null4 [new Agent/Null]
$ns attach-agent $n18 $null4
$ns connect $udp4 $null4
```

```
$udp4 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4
$cbr4 set packetSize_ 1000
$cbr4 set rate_ 0.1Mb
$cbr4 set random_ null
$ns at 60.0 "$cbr4 start"
$ns at 80.0 "$cbr4 stop"
set udp5 [new Agent/UDP]
$ns attach-agent $n16 $udp5
set null5 [new Agent/Null]
$ns attach-agent $n18 $null5
$ns connect $udp5 $null5
$udp5 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp5
$cbr5 set packetSize_ 1000
$cbr5 set rate_ 0.1Mb
$cbr5 set random_ null
$ns at 80.0 "$cbr5 start"
$ns at 100.0 "$cbr5 stop"

#====================================
#        Termination
#====================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam normal.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

## Blackhole Attack Code

blackholeattack.tcl

```
#===================================
#     Simulation parameters setup
#===================================
set val(chan)   Channel/WirelessChannel   ;# channel type
set val(prop)   Propagation/TwoRayGround  ;# propagation model
set val(netif)  Phy/WirelessPhy           ;# network interface type
set val(mac)    Mac/802_11                ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue   ;# interface queue type
set val(ll)     LL                        ;# link layer type
set val(ant)    Antenna/OmniAntenna       ;# antenna model
set val(ifqlen) 50                        ;# max packet in ifq
set val(nn)     25                 ;# number of mobilenodes
set val(rp)     AODV               ;# routing protocol
set val(x)      1200               ;# X dimension of topography
set val(y)      600                ;# Y dimension of topography
set val(stop)   120               ;# time of simulation end (s)
set val(energymodel)    EnergyModel     ;
set val(radiomodel)     RadioModel      ;
set val(initialenergy)  1000           ;# Initial energy (J)

#===================================
#        Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open blackhole.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open blackhole.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];# Create wireless channel
```

```
#====================================
#     Mobile node parameter setup
#====================================
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      OFF \
                -movementTrace ON \
                -energyModel $val(energymodel) \
                -idlePower 1.0 \
                -rxPower 1.0 \
                -txPower 2.0 \
                -sleepPower 0.001 \
                -transitionPower 0.2 \
                -transitionTime 0.005 \
                -initialEnergy $val(initialenergy)


#====================================
#        Nodes Definition
#====================================
#Create 25 nodes
set n0 [$ns node]
$n0 set X_ 663
$n0 set Y_ 484
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 466
$n1 set Y_ 407
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 871
$n2 set Y_ 426
$n2 set Z_ 0.0
```

```
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 668
$n3 set Y_ 393
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 558
$n4 set Y_ 320
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 781
$n5 set Y_ 317
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 523
$n6 set Y_ 222
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 671
$n7 set Y_ 194
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20
set n8 [$ns node]
$n8 set X_ 891
$n8 set Y_ 224
$n8 set Z_ 0.0
$ns initial_node_pos $n8 20
set n9 [$ns node]
$n9 set X_ 476
$n9 set Y_ 117
$n9 set Z_ 0.0
$ns initial_node_pos $n9 20
set n10 [$ns node]
$n10 set X_ 674
$n10 set Y_ 112
$n10 set Z_ 0.0
$ns initial_node_pos $n10 20
set n11 [$ns node]
$n11 set X_ 895
$n11 set Y_ 130
$n11 set Z_ 0.0
```

```
$ns initial_node_pos $n11 20
set n12 [$ns node]
$n12 set X_ 500
$n12 set Y_ 300
$n12 set Z_ 0.0
$ns initial_node_pos $n12 20
set n13 [$ns node]
$n13 set X_ 687
$n13 set Y_ 36
$n13 set Z_ 0.0
$ns initial_node_pos $n13 20
set n14 [$ns node]
$n14 set X_ 877
$n14 set Y_ 39
$n14 set Z_ 0.0
$ns initial_node_pos $n14 20
set n15 [$ns node]
$n15 set X_ 373
$n15 set Y_ 271
$n15 set Z_ 0.0
$ns initial_node_pos $n15 20
set n16 [$ns node]
$n16 set X_ 990
$n16 set Y_ 306
$n16 set Z_ 0.0
$ns initial_node_pos $n16 20
set n17 [$ns node]
$n17 set X_ 989
$n17 set Y_ 407
$n17 set Z_ 0.0
$ns initial_node_pos $n17 20
set n18 [$ns node]
$n18 set X_ 1086
$n18 set Y_ 453
$n18 set Z_ 0.0
$ns initial_node_pos $n18 20
set n19 [$ns node]
$n19 set X_ 455
$n19 set Y_ 479
$n19 set Z_ 0.0
$ns initial_node_pos $n19 20
set n20 [$ns node]
$n20 set X_ 350
$n20 set Y_ 434
$n20 set Z_ 0.0
```

```
$ns initial_node_pos $n20 20
set n21 [$ns node]
$n21 set X_ 263
$n21 set Y_ 306
$n21 set Z_ 0.0
$ns initial_node_pos $n21 20
set n22 [$ns node]
$n22 set X_ 261
$n22 set Y_ 209
$n22 set Z_ 0.0
$ns initial_node_pos $n22 20
set n23 [$ns node]
$n23 set X_ 240
$n23 set Y_ 115
$n23 set Z_ 0.0
$ns initial_node_pos $n23 20
set n24 [$ns node]
$n24 set X_ 313
$n24 set Y_ 29
$n24 set Z_ 0.0
$ns initial_node_pos $n24 20

#===================================
#        Generate movement
#===================================
$ns at 0 " $n6 setdest 1086 453 40 "
$ns at 10 " $n18 setdest 877 39 40 "
$ns at 20 " $n18 setdest 500 117 40 "
$ns at 60 " $n18 setdest 400 100 40 "
$ns at 40 " $n6 setdest 400 500 40 "
$ns at 10 " $n15 setdest 650 470 40 "
$ns at 10 " $n5 setdest 550 220 40 "

#Blackhole attackers
$ns at 0.0 "[$n7 set ragent_] blackhole"
$ns at 0.0 "[$n8 set ragent_] blackhole"
$ns at 0.0 "[$n10 set ragent_] blackhole"

#===================================
#        Agents Definition
#===================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n21 $udp0
set null1 [new Agent/Null]
```

```
$ns attach-agent $n18 $null1
$ns connect $udp0 $null1
$udp0 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 0.1Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 20.0 "$cbr0 stop"
#Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n20 $udp1
set null2 [new Agent/Null]
$ns attach-agent $n18 $null2
$ns connect $udp1 $null1
$udp1 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 0.1Mb
$cbr1 set random_ null
$ns at 20.0 "$cbr1 start"
$ns at 40.0 "$cbr1 stop"
#Setup a UDP connection
set udp3 [new Agent/UDP]
$ns attach-agent $n22 $udp3
set null3 [new Agent/Null]
$ns attach-agent $n18 $null3
$ns connect $udp3 $null1
$udp3 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp3
$cbr2 set packetSize_ 1000
$cbr2 set rate_ 0.1Mb
$cbr2 set random_ null
$ns at 40.0 "$cbr2 start"
$ns at 60.0 "$cbr2 stop"
set udp4 [new Agent/UDP]
```

```
$ns attach-agent $n8 $udp4
set null4 [new Agent/Null]
$ns attach-agent $n18 $null4
$ns connect $udp4 $null4
$udp4 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4
$cbr4 set packetSize_ 1000
$cbr4 set rate_ 0.1Mb
$cbr4 set random_ null
$ns at 60.0 "$cbr4 start"
$ns at 80.0 "$cbr4 stop"
set udp5 [new Agent/UDP]
$ns attach-agent $n16 $udp5
set null5 [new Agent/Null]
$ns attach-agent $n18 $null5
$ns connect $udp5 $null5
$udp5 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp5
$cbr5 set packetSize_ 1000
$cbr5 set rate_ 0.1Mb
$cbr5 set random_ null
$ns at 80.0 "$cbr5 start"
$ns at 100.0 "$cbr5 stop"

#===================================
#        Termination
#===================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam blackhole.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
```

```
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

## Rushing Attack Code

### rushingattack.tcl

```
#===================================
#     Simulation parameters setup
#===================================
set val(chan)   Channel/WirelessChannel   ;# channel type
set val(prop)   Propagation/TwoRayGround  ;# propagation model
set val(netif)  Phy/WirelessPhy           ;# network interface type
set val(mac)    Mac/802_11                ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue   ;# interface queue type
set val(ll)     LL                        ;# link layer type
set val(ant)    Antenna/OmniAntenna       ;# antenna model
set val(ifqlen) 50                        ;# max packet in ifq
set val(nn)     25                 ;# number of mobilenodes
set val(rp)     AODV               ;# routing protocol
set val(x)      1200               ;# X dimension of topography
set val(y)      600                ;# Y dimension of topography
set val(stop)   120               ;# time of simulation end (s)
set val(energymodel)    EnergyModel     ;
set val(radiomodel)     RadioModel      ;
set val(initialenergy)  1000           ;# Initial energy (J)

#===================================
#        Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open rushing.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open rushing.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];# Create wireless channel
#===================================
#     Mobile node parameter setup
```

```
#====================================
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      ON \
                -movementTrace ON \
                -energyModel $val(energymodel) \
                -idlePower 1.0 \
                -rxPower 1.0 \
                -txPower 2.0 \
                -sleepPower 0.001 \
                -transitionPower 0.2 \
                -transitionTime 0.005 \
                -initialEnergy $val(initialenergy)


#====================================
#        Nodes Definition
#====================================
#Create 25 nodes
set n0 [$ns node]
$n0 set X_ 663
$n0 set Y_ 484
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 466
$n1 set Y_ 407
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 871
$n2 set Y_ 426
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
```

```
$n3 set X_ 668
$n3 set Y_ 393
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 558
$n4 set Y_ 320
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 781
$n5 set Y_ 317
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 523
$n6 set Y_ 222
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 671
$n7 set Y_ 194
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20
set n8 [$ns node]
$n8 set X_ 891
$n8 set Y_ 224
$n8 set Z_ 0.0
$ns initial_node_pos $n8 20
set n9 [$ns node]
$n9 set X_ 476
$n9 set Y_ 117
$n9 set Z_ 0.0
$ns initial_node_pos $n9 20
set n10 [$ns node]
$n10 set X_ 674
$n10 set Y_ 112
$n10 set Z_ 0.0
$ns initial_node_pos $n10 20
set n11 [$ns node]
$n11 set X_ 895
$n11 set Y_ 130
$n11 set Z_ 0.0
$ns initial_node_pos $n11 20
set n12 [$ns node]
```

```
$n12 set X_ 500
$n12 set Y_ 300
$n12 set Z_ 0.0
$ns initial_node_pos $n12 20
set n13 [$ns node]
$n13 set X_ 687
$n13 set Y_ 36
$n13 set Z_ 0.0
$ns initial_node_pos $n13 20
set n14 [$ns node]
$n14 set X_ 877
$n14 set Y_ 39
$n14 set Z_ 0.0
$ns initial_node_pos $n14 20
set n15 [$ns node]
$n15 set X_ 373
$n15 set Y_ 271
$n15 set Z_ 0.0
$ns initial_node_pos $n15 20
set n16 [$ns node]
$n16 set X_ 990
$n16 set Y_ 306
$n16 set Z_ 0.0
$ns initial_node_pos $n16 20
set n17 [$ns node]
$n17 set X_ 989
$n17 set Y_ 407
$n17 set Z_ 0.0
$ns initial_node_pos $n17 20
set n18 [$ns node]
$n18 set X_ 1086
$n18 set Y_ 453
$n18 set Z_ 0.0
$ns initial_node_pos $n18 20
set n19 [$ns node]
$n19 set X_ 455
$n19 set Y_ 479
$n19 set Z_ 0.0
$ns initial_node_pos $n19 20
set n20 [$ns node]
$n20 set X_ 350
$n20 set Y_ 434
$n20 set Z_ 0.0
$ns initial_node_pos $n20 20
set n21 [$ns node]
```

```
$n21 set X_ 263
$n21 set Y_ 306
$n21 set Z_ 0.0
$ns initial_node_pos $n21 20
set n22 [$ns node]
$n22 set X_ 261
$n22 set Y_ 209
$n22 set Z_ 0.0
$ns initial_node_pos $n22 20
set n23 [$ns node]
$n23 set X_ 240
$n23 set Y_ 115
$n23 set Z_ 0.0
$ns initial_node_pos $n23 20
set n24 [$ns node]
$n24 set X_ 313
$n24 set Y_ 29
$n24 set Z_ 0.0
$ns initial_node_pos $n24 20


#===================================
#        Generate movement
#===================================
$ns at 0 " $n6 setdest 1086 453 40 "
$ns at 10 " $n18 setdest 877 39 40 "
$ns at 20 " $n18 setdest 500 117 40 "
$ns at 60 " $n18 setdest 400 100 40 "
$ns at 40 " $n6 setdest 400 500 40 "
$ns at 10 " $n15 setdest 650 470 40 "
$ns at 10 " $n5 setdest 550 220 40 "
#$ns at 40 " $n0 t1 "

#Rushing attackers
$ns at 0.0 "[$n7 set ragent_] rushingattack"
$ns at 0.0 "[$n8 set ragent_] rushingattack"
$ns at 0.0 "[$n10 set ragent_] rushingattack"

#===================================
#        Agents Definition
#===================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n21 $udp0
set null1 [new Agent/Null]
```

```
$ns attach-agent $n18 $null1
$ns connect $udp0 $null1
$udp0 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 0.1Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 20.0 "$cbr0 stop"
#Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n20 $udp1
set null2 [new Agent/Null]
$ns attach-agent $n18 $null2
$ns connect $udp1 $null1
$udp1 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 0.1Mb
$cbr1 set random_ null
$ns at 20.0 "$cbr1 start"
$ns at 40.0 "$cbr1 stop"
#Setup a UDP connection
set udp3 [new Agent/UDP]
$ns attach-agent $n22 $udp3
set null3 [new Agent/Null]
$ns attach-agent $n18 $null3
$ns connect $udp3 $null1
$udp3 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp3
$cbr2 set packetSize_ 1000
$cbr2 set rate_ 0.1Mb
$cbr2 set random_ null
$ns at 40.0 "$cbr2 start"
$ns at 60.0 "$cbr2 stop"
set udp4 [new Agent/UDP]
```

```
$ns attach-agent $n8 $udp4
set null4 [new Agent/Null]
$ns attach-agent $n18 $null4
$ns connect $udp4 $null4
$udp4 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4
$cbr4 set packetSize_ 1000
$cbr4 set rate_ 0.1Mb
$cbr4 set random_ null
$ns at 60.0 "$cbr4 start"
$ns at 80.0 "$cbr4 stop"
set udp5 [new Agent/UDP]
$ns attach-agent $n16 $udp5
set null5 [new Agent/Null]
$ns attach-agent $n18 $null5
$ns connect $udp5 $null5
$udp5 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp5
$cbr5 set packetSize_ 1000
$cbr5 set rate_ 0.1Mb
$cbr5 set random_ null
$ns at 80.0 "$cbr5 start"
$ns at 100.0 "$cbr5 stop"

#====================================
#        Termination
#====================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam rushing.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
```

```
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

## Flooding Attack Code

### floodingattack.tcl

```
#====================================
#     Simulation parameters setup
#====================================
set val(chan)   Channel/WirelessChannel   ;# channel type
set val(prop)   Propagation/TwoRayGround  ;# propagation model
set val(netif)  Phy/WirelessPhy           ;# network interface type
set val(mac)    Mac/802_11                ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue   ;# interface queue type
set val(ll)     LL                        ;# link layer type
set val(ant)    Antenna/OmniAntenna       ;# antenna model
set val(ifqlen) 50                        ;# max packet in ifq
set val(nn)     25                 ;# number of mobilenodes
set val(rp)     AODV               ;# routing protocol
set val(x)      1200               ;# X dimension of topography
set val(y)      600                ;# Y dimension of topography
set val(stop)   120               ;# time of simulation end (s)
set val(energymodel)    EnergyModel     ;
set val(radiomodel)     RadioModel      ;
set val(initialenergy)  1000            ;# Initial energy (J)

#====================================
#        Initialization
#====================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open flooding.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open flooding.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];# Create wireless channel
#====================================
#     Mobile node parameter setup
```

```
#===================================
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      OFF \
                -movementTrace ON \
                -energyModel $val(energymodel) \
                -idlePower 1.0 \
                -rxPower 1.0 \
                -txPower 2.0 \
                -sleepPower 0.001 \
                -transitionPower 0.2 \
                -transitionTime 0.005 \
                -initialEnergy $val(initialenergy)


#===================================
#        Nodes Definition
#===================================
#Create 25 nodes
set n0 [$ns node]
$n0 set X_ 663
$n0 set Y_ 484
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 466
$n1 set Y_ 407
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 871
$n2 set Y_ 426
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
```

```
$n3 set X_ 668
$n3 set Y_ 393
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 558
$n4 set Y_ 320
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 781
$n5 set Y_ 317
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 523
$n6 set Y_ 222
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 671
$n7 set Y_ 194
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20
set n8 [$ns node]
$n8 set X_ 891
$n8 set Y_ 224
$n8 set Z_ 0.0
$ns initial_node_pos $n8 20
set n9 [$ns node]
$n9 set X_ 476
$n9 set Y_ 117
$n9 set Z_ 0.0
$ns initial_node_pos $n9 20
set n10 [$ns node]
$n10 set X_ 674
$n10 set Y_ 112
$n10 set Z_ 0.0
$ns initial_node_pos $n10 20
set n11 [$ns node]
$n11 set X_ 895
$n11 set Y_ 130
$n11 set Z_ 0.0
$ns initial_node_pos $n11 20
set n12 [$ns node]
```

```
$n12 set X_ 500
$n12 set Y_ 300
$n12 set Z_ 0.0
$ns initial_node_pos $n12 20
set n13 [$ns node]
$n13 set X_ 687
$n13 set Y_ 36
$n13 set Z_ 0.0
$ns initial_node_pos $n13 20
set n14 [$ns node]
$n14 set X_ 877
$n14 set Y_ 39
$n14 set Z_ 0.0
$ns initial_node_pos $n14 20
set n15 [$ns node]
$n15 set X_ 373
$n15 set Y_ 271
$n15 set Z_ 0.0
$ns initial_node_pos $n15 20
set n16 [$ns node]
$n16 set X_ 990
$n16 set Y_ 306
$n16 set Z_ 0.0
$ns initial_node_pos $n16 20
set n17 [$ns node]
$n17 set X_ 989
$n17 set Y_ 407
$n17 set Z_ 0.0
$ns initial_node_pos $n17 20
set n18 [$ns node]
$n18 set X_ 1086
$n18 set Y_ 453
$n18 set Z_ 0.0
$ns initial_node_pos $n18 20
set n19 [$ns node]
$n19 set X_ 455
$n19 set Y_ 479
$n19 set Z_ 0.0
$ns initial_node_pos $n19 20
set n20 [$ns node]
$n20 set X_ 350
$n20 set Y_ 434
$n20 set Z_ 0.0
$ns initial_node_pos $n20 20
set n21 [$ns node]
```

```
$n21 set X_ 263
$n21 set Y_ 306
$n21 set Z_ 0.0
$ns initial_node_pos $n21 20
set n22 [$ns node]
$n22 set X_ 261
$n22 set Y_ 209
$n22 set Z_ 0.0
$ns initial_node_pos $n22 20
set n23 [$ns node]
$n23 set X_ 240
$n23 set Y_ 115
$n23 set Z_ 0.0
$ns initial_node_pos $n23 20
set n24 [$ns node]
$n24 set X_ 313
$n24 set Y_ 29
$n24 set Z_ 0.0
$ns initial_node_pos $n24 20


#===================================
#        Generate movement
#===================================
$ns at 0 " $n6 setdest 1086 453 40 "
$ns at 10 " $n18 setdest 877 39 40 "
$ns at 20 " $n18 setdest 500 117 40 "
$ns at 60 " $n18 setdest 400 100 40 "
$ns at 40 " $n6 setdest 400 500 40 "
$ns at 10 " $n15 setdest 650 470 40 "
$ns at 10 " $n5 setdest 550 220 40 "


#Flooding attackers
$ns at 0.0 "[$n7 set ragent_] flooder"
$ns at 0.0 "[$n8 set ragent_] flooder"
$ns at 0.0 "[$n10 set ragent_] flooder"

#===================================
#        Agents Definition
#===================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n21 $udp0
set null1 [new Agent/Null]
```

```
$ns attach-agent $n18 $null1
$ns connect $udp0 $null1
$udp0 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 0.1Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 20.0 "$cbr0 stop"
#Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n20 $udp1
set null2 [new Agent/Null]
$ns attach-agent $n18 $null2
$ns connect $udp1 $null1
$udp1 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 0.1Mb
$cbr1 set random_ null
$ns at 20.0 "$cbr1 start"
$ns at 40.0 "$cbr1 stop"
#Setup a UDP connection
set udp3 [new Agent/UDP]
$ns attach-agent $n22 $udp3
set null3 [new Agent/Null]
$ns attach-agent $n18 $null3
$ns connect $udp3 $null1
$udp3 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp3
$cbr2 set packetSize_ 1000
$cbr2 set rate_ 0.1Mb
$cbr2 set random_ null
$ns at 40.0 "$cbr2 start"
$ns at 60.0 "$cbr2 stop"
set udp4 [new Agent/UDP]
```

```
$ns attach-agent $n8 $udp4
set null4 [new Agent/Null]
$ns attach-agent $n18 $null4
$ns connect $udp4 $null4
$udp4 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4
$cbr4 set packetSize_ 1000
$cbr4 set rate_ 0.1Mb
$cbr4 set random_ null
$ns at 60.0 "$cbr4 start"
$ns at 80.0 "$cbr4 stop"
set udp5 [new Agent/UDP]
$ns attach-agent $n16 $udp5
set null5 [new Agent/Null]
$ns attach-agent $n18 $null5
$ns connect $udp5 $null5
$udp5 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp5
$cbr5 set packetSize_ 1000
$cbr5 set rate_ 0.1Mb
$cbr5 set random_ null
$ns at 80.0 "$cbr5 start"
$ns at 100.0 "$cbr5 stop"

#===================================
#        Termination
#===================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam flooding.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
```

```
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

## Selective Forwarding Attack Code

### grayholeattack.tcl

```
#====================================
#     Simulation parameters setup
#====================================
set val(chan)   Channel/WirelessChannel   ;# channel type
set val(prop)   Propagation/TwoRayGround  ;# propagation model
set val(netif)  Phy/WirelessPhy           ;# network interface type
set val(mac)    Mac/802_11                ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue   ;# interface queue type
set val(ll)     LL                        ;# link layer type
set val(ant)    Antenna/OmniAntenna       ;# antenna model
set val(ifqlen) 50                        ;# max packet in ifq
set val(nn)     25                 ;# number of mobilenodes
set val(rp)     AODV               ;# routing protocol
set val(x)      1200               ;# X dimension of topography
set val(y)      600                ;# Y dimension of topography
set val(stop)   120               ;# time of simulation end (s)
set val(energymodel)    EnergyModel     ;
set val(radiomodel)     RadioModel      ;
set val(initialenergy)  1000           ;# Initial energy (J)

#====================================
#        Initialization
#====================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open grayhole.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open grayhole.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];# Create wireless channel
#====================================
#     Mobile node parameter setup
```

```
#====================================
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      OFF \
                -movementTrace ON \
                -energyModel $val(energymodel) \
                -idlePower 1.0 \
                -rxPower 1.0 \
                -txPower 2.0 \
                -sleepPower 0.001 \
                -transitionPower 0.2 \
                -transitionTime 0.005 \
                -initialEnergy $val(initialenergy)


#====================================
#        Nodes Definition
#====================================
#Create 25 nodes
set n0 [$ns node]
$n0 set X_ 663
$n0 set Y_ 484
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 466
$n1 set Y_ 407
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 871
$n2 set Y_ 426
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
```

```
$n3 set X_ 668
$n3 set Y_ 393
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 558
$n4 set Y_ 320
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 781
$n5 set Y_ 317
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 523
$n6 set Y_ 222
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 671
$n7 set Y_ 194
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20
set n8 [$ns node]
$n8 set X_ 891
$n8 set Y_ 224
$n8 set Z_ 0.0
$ns initial_node_pos $n8 20
set n9 [$ns node]
$n9 set X_ 476
$n9 set Y_ 117
$n9 set Z_ 0.0
$ns initial_node_pos $n9 20
set n10 [$ns node]
$n10 set X_ 674
$n10 set Y_ 112
$n10 set Z_ 0.0
$ns initial_node_pos $n10 20
set n11 [$ns node]
$n11 set X_ 895
$n11 set Y_ 130
$n11 set Z_ 0.0
$ns initial_node_pos $n11 20
set n12 [$ns node]
```

```
$n12 set X_ 500
$n12 set Y_ 300
$n12 set Z_ 0.0
$ns initial_node_pos $n12 20
set n13 [$ns node]
$n13 set X_ 687
$n13 set Y_ 36
$n13 set Z_ 0.0
$ns initial_node_pos $n13 20
set n14 [$ns node]
$n14 set X_ 877
$n14 set Y_ 39
$n14 set Z_ 0.0
$ns initial_node_pos $n14 20
set n15 [$ns node]
$n15 set X_ 373
$n15 set Y_ 271
$n15 set Z_ 0.0
$ns initial_node_pos $n15 20
set n16 [$ns node]
$n16 set X_ 990
$n16 set Y_ 306
$n16 set Z_ 0.0
$ns initial_node_pos $n16 20
set n17 [$ns node]
$n17 set X_ 989
$n17 set Y_ 407
$n17 set Z_ 0.0
$ns initial_node_pos $n17 20
set n18 [$ns node]
$n18 set X_ 1086
$n18 set Y_ 453
$n18 set Z_ 0.0
$ns initial_node_pos $n18 20
set n19 [$ns node]
$n19 set X_ 455
$n19 set Y_ 479
$n19 set Z_ 0.0
$ns initial_node_pos $n19 20
set n20 [$ns node]
$n20 set X_ 350
$n20 set Y_ 434
$n20 set Z_ 0.0
$ns initial_node_pos $n20 20
set n21 [$ns node]
```

```
$n21 set X_ 263
$n21 set Y_ 306
$n21 set Z_ 0.0
$ns initial_node_pos $n21 20
set n22 [$ns node]
$n22 set X_ 261
$n22 set Y_ 209
$n22 set Z_ 0.0
$ns initial_node_pos $n22 20
set n23 [$ns node]
$n23 set X_ 240
$n23 set Y_ 115
$n23 set Z_ 0.0
$ns initial_node_pos $n23 20
set n24 [$ns node]
$n24 set X_ 313
$n24 set Y_ 29
$n24 set Z_ 0.0
$ns initial_node_pos $n24 20


#===================================
#        Generate movement
#===================================
$ns at 0 " $n6 setdest 1086 453 40 "
$ns at 10 " $n18 setdest 877 39 40 "
$ns at 20 " $n18 setdest 500 117 40 "
$ns at 60 " $n18 setdest 400 100 40 "
$ns at 40 " $n6 setdest 400 500 40 "
$ns at 10 " $n15 setdest 650 470 40 "
$ns at 10 " $n5 setdest 550 220 40 "


#Flooding attackers
$ns at 0.0 "[$n7 set ragent_] grayhole"
$ns at 0.0 "[$n8 set ragent_] grayhole"
$ns at 0.0 "[$n10 set ragent_] grayhole"

#===================================
#        Agents Definition
#===================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n21 $udp0
set null1 [new Agent/Null]
```

```
$ns attach-agent $n18 $null1
$ns connect $udp0 $null1
$udp0 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 0.1Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 20.0 "$cbr0 stop"
#Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n20 $udp1
set null2 [new Agent/Null]
$ns attach-agent $n18 $null2
$ns connect $udp1 $null1
$udp1 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 0.1Mb
$cbr1 set random_ null
$ns at 20.0 "$cbr1 start"
$ns at 40.0 "$cbr1 stop"
#Setup a UDP connection
set udp3 [new Agent/UDP]
$ns attach-agent $n22 $udp3
set null3 [new Agent/Null]
$ns attach-agent $n18 $null3
$ns connect $udp3 $null1
$udp3 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp3
$cbr2 set packetSize_ 1000
$cbr2 set rate_ 0.1Mb
$cbr2 set random_ null
$ns at 40.0 "$cbr2 start"
$ns at 60.0 "$cbr2 stop"
set udp4 [new Agent/UDP]
```

```
$ns attach-agent $n8 $udp4
set null4 [new Agent/Null]
$ns attach-agent $n18 $null4
$ns connect $udp4 $null4
$udp4 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4
$cbr4 set packetSize_ 1000
$cbr4 set rate_ 0.1Mb
$cbr4 set random_ null
$ns at 60.0 "$cbr4 start"
$ns at 80.0 "$cbr4 stop"
set udp5 [new Agent/UDP]
$ns attach-agent $n16 $udp5
set null5 [new Agent/Null]
$ns attach-agent $n18 $null5
$ns connect $udp5 $null5
$udp5 set packetSize_ 1500

#Setup a CBR Application over UDP connection
set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp5
$cbr5 set packetSize_ 1000
$cbr5 set rate_ 0.1Mb
$cbr5 set random_ null
$ns at 80.0 "$cbr5 start"
$ns at 100.0 "$cbr5 stop"

#===================================
#        Termination
#===================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam grayhole.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
```

```
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

## Generic Attack Code

### aodv.h

```
/*
    The Routing Agent
*/
class AODV: public Agent
{
   ..........

 /*
  * History management
  */
    bool   malicious;
       ..........
 }
```

### aodv.cc

```
AODV::AODV(nsaddr_t id) : Agent(PT_AODV), btimer(this), htimer(this),
ntimer(this), rtimer(this), lrtimer(this), rqueue(){

            index = id;
            seqno = 2;
            bid = 1;

            malicious=false;
             ............
         }
```

---

```
int AODV::command(int argc, const char*const* argv)
{
    if(argc == 2)
   {
         Tcl& tcl = Tcl::instance();

        if(strncasecmp(argv[1], "id", 2) == 0)
        {
              tcl.resultf("%d", index);
               return TCL_OK;
        }

        if(strcmp(argv[1], "hacker") == 0)
        {
                malicious=true;
                return TCL_OK;
        }
             ...........
      }
 }
```

```
                    /*
                       Route Handling Functions
                    */

                    void AODV::rt_resolve(Packet *p)
                    {
                       ..........
                      // If i am a malicious node,
                      if (malicious == true)
                     {
                               drop(p, DROP_RTR_NO_ROUTE);
                               // DROP_RTR_NO_ROUTE is added for no reason
                               return;    //Required if you get pkt flow
not specified error.
                      }
                       .........
                     }
```

**aodv.tcl**

In TCL file we will create we have to add the following line after packet transmission.

```
        $ns at 0.0 "[$node_(0) set ragent_] hacker"
```