الجمـــهوريـــة الجـــزائريـــة الديـمقـراطيـــــــة الشـــعبيـــــة
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعـلـيـم الـعـالـــي و الـبـحـث الـعـلـمـي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
جـامـعـة عمّـــار ثـلـيـــجـــي بالأغـــواط
UNIVERSITE AMAR TELIDJI LAGHOUAT
كـلـيـــــة الـعـلـــــوم
FACULTE DES SCIENCES
DEPARTEMENT DE MATHEMATIQUES ET INFORMATIQUE

Mémoire de MASTER

Domaine : Mathématiques et Informatique
Filière : Informatiques
Option : Réseaux, Systèmes et Applications Réparties

Par:
Mohamed Riadh BENABDELKARIM

THEME

Intrusion Detection System for Wireless Sensor Network

Soutenu publiquement le 12-06-2017 devant le jury composé de:

Mr Noureddine CHAIB	*M.A.(A)*	*Président*
Mr Younes GUELLOUMA	*M.C.(A)*	*Examinateur*
Mme Hadda CHERROUN	*Professeur*	*Encadreur*
Mr Khaled BEKKAR	*M.A.(A)*	*Co-encadreur*

Année Universitaire 2016/2017

Dedication

I dedicate this thesis to :

- My family with gratitude to my loving parents, and their words of encouragement and push for tenancy.
- My dear friends Khalil, Hakuibe, Jihed, Mimo, and Rafik who have never left my side on cloudy days.
- My wonderful supervisor Hadda Cherroun for the many hours of proofreading throughout the entire writing of this thesis.
- My co-supervisor Khaled Bekkar for helping me refine my programming skills.
- my colleagues and brothers.

All of you have been my best supporters.

Benabdelkrim Mohamed Riadh...

Acknowledgements

Firstly, all praise and thanks to Allah for his grace, guidance, and giving us the power to finish this work.

I thank :

- Ms.Hadda Cherroun for her guidance and help to finish this thesis
- Mr.Khaled Bekkar for his assistance in both coding and thesis writing.
- Jihed and Khalil for their moral support. and Finally all my friends and colleagues.
- my colleagues and brothers.

Benabdelkarim Mohamed Riadh...

ملخص

إن لشبكات الاستشعار اللاسلكية (WSN) العديد من التطبيقات في كل المجالات تقريبا. وهي تتكون من أجهزة صغيرة رخيصة الثمن و التي يتم نشرها في بيئات مفتوحة وغير آمنة, مما يجعلها عرضة لجميع أنواع الهجمات . لذا فإن تشغيل هده الشبكات بشكل آمن يكتسي أهمية كبيرة. ومن اجل ذلك، ينبغي التعرف على أي اختراق قبل أن تتضرر الشبكة من طرف أي هجوم. من بين حلول أنظمة كشف التسلل (IDS)، يستعمل نظام كشف التسلل القائم على التعلم والتي قد أثبتت كفاءتها .ومع ذلك، فان النظام القائم على التعلم يحتاج إلى مجموعة من الميزات لتوصيف الهجمات مع احترام خصوصيات الشبكات (WSN).

في إطار هذا المشروع، قمنا بالتحقيق في ما تتسم به الهجمات على النحو الأمثل وذلك بناءا على 6568 محاكاة ، حيث حصلنا على مجموعة بيانات واسعة تتكون من 94.426 تسجيل مختلف يصف سلوكيات الاتصال. والسلوكيات المستهدفة هي العادية، الهوة السوداء، Hello-Flood و Dos.استنادا إلى الأداء التصنيفي، حيث أثبتت نتائجنا على مستوى الوصلات، أن الهجمات يمكن أن تتميز بأربعة سمات فقط. عن طريق تصنيف الغابات العشوائي، حصلنا على 91.8 % من الدقة في السيناريوهات المنخفضة العبئ و 78.1 % في السيناريوهات العالية العبئ.

الكلمات المفتاحية:
شبكات الاستشعار اللاسلكية (WSN)، نظام كشف التسلل (IDS)، المحاكاة، التعلم الآلي، ميزات، الحرمان من الخدمة، هوة سوداء، Hello-Flood، DoS، الغابات العشوائية.

Abstract

Wireless sensor networks (WSNs) have numerous application in almost every domain. WSNs are composed of cheap tiny devices that are deployed in open and unsafe environments making them exposed to all sorts of attacks. The security of such networks is of great importance. Hence, securely operating WSNs, any intrusions should be recognized before attackers can harm the network. Among the IDS solution, Machine learning-based IDSs have proved their efficiency. However, Machine learning-based IDS for WSN needs a set of features to characterize attacks while respecting WSN specificities.

In this work, we investigate how to sub-optimally characterize attacks in WSN. Based on 6,568 deployed simulations, we have got a large dataset of 94,426 records that captures different behaviors at the connection level. The targeted behaviors are Normal, Blackhole, Hello-Flood, and DoS. Based on classification performances, our results prove that at the connection level, attacks can be characterized by just four attributes. Using Random Forest classifier, we reached 91.8% of precision in the low-loaded scenario and 78.1% in the high-loaded scenarios.

keywords: Wireless Sensor Network (WSN), Intrusion Detection System (IDS), Simulations, Machine learning, Features, Denial of Service, Blackhole, Hello-Flood, Random Forest.

Résumé

Les réseaux de capteurs sans fil (WSN) ont de nombreuses applications dans presque tous les domaines. Les WSN sont composés de petits périphériques peu coûteux qui sont déployés dans des environnements ouverts et dangereux, ce qui les rend exposés à toutes sortes d'attaques. La sécurité de ces réseaux revêt une grande importance. Par conséquent, les WSN sécurisés, toutes les intrusions devraient être reconnues avant que les attaquants ne nuisent au réseau. Parmi les solutions IDS (Intrusion Detection System), les IDS basés sur l'apprentissage de la machine ont prouvé leur efficacité. Cependant, l'IDS basé sur l'apprentissage par machine (Machine learning-based IDSs) pour WSN nécessite un ensemble de fonctionnalités pour caractériser les attaques tout en respectant les spécificités WSN. Dans ce travail, nous étudions comment caractériser de manière optimale les attaques dans WSN. Sur la base de 6 568 simulations déployées, nous avons un grand ensemble de données de 94 426 enregistrements qui caractérisent les différents comportements au niveau de la connexion. Les comportements ciblés sont Normal, Blackhole, Hello-Flood et DoS. Sur la base des performances de classement, nos résultats prouvent qu'au niveau de la connexion, les attaques peuvent être caractérisées par seulement quatre attributs. En utilisant le classificateur par forêt aléatoire, nous avons atteint $91,8$ % de précision dans le scénario à faible charge et $78,1$ % dans les scénarios à haute charge.

mots-clés: Réseaux de capteurs sans fil (WSN), Système de détection d'intrusion (IDS), Simulations, apprentissage automatique, fonctionnalités, déni de service, Blackhole, Hello-Flood, forêt aléatoire.

Table of Contents

List of Figures

List of Tables

Introduction

The progress humanity has achieved in wireless communications and electronics have allowed us to develop low-cost multifunctional sensor nodes. These sensor nodes are battery powered, low-priced, tiny and are capable of performing some processing, accumulating sensed data and communicating wirelessly with other connected nodes a network known as Wireless Sensor Network (WSN). Despite their small size, sensor nodes have endless possible applications in every sector. In the military sector, for example, they can be used imaging fields and tracking individuals, vehicles, and equipments or in more delicate applications such as chemical, biological, radiological, nuclear and explosives [59]. Advances in wireless sensor networking have also opened up new possibilities in healthcare systems. WSNs technology has overrun medical instruments, replacing thousands of wires connected to hospital devices. This technology also has the capacity of providing reliability in addition to enhanced mobility [46].

Nonetheless, WSNs are vulnerable to numerous kinds of attacks [36, 63] like node capturing, eavesdropping, Man in the middle, and even falsifying sent data [26]. Due to the delicacy of many WSN applications [15], WSNs security is of critical importance, while there are many mechanics to prevent attacks [35, 51] there are several attacks for which there are no

perceived prevention methods, for instance, wormhole [29, 36]. Furthermore, preventive methods do not guarantee to hold intruders. For these cases, it is essential to adopt some mechanism of intrusion detection. In addition to deterring the intruder from crippling the network, the intrusion detection system (IDS) can identify both known or unknown attacks [57].

Although the prerequisites of an IDS for both wired and wireless networks are similar, WSN imposes additional challenges. Overall, the effectiveness of solutions designed for wired networks is restricted for WSNs regarding energy, memory, and computing power [13].

IDSs are classified into three categories according to their detection methodology: Misuse-based, Specification-based, and Anomaly-based detection method [14]. This latter has several detection techniques: Statistical-based, Knowledge-based, and Machine Learning-based detection.

Machine learning-based IDS has proven its efficiency [58, 64]. It can distinguish anomalies from normal behaviors by using a set of features that can characterize different behaviors.

In this work, we are challenged with the task of finding a sub-optimal set of features to build a Machine learning IDS that respects WSN limitations in terms of energy, memory, and processing power. The document contains three chapters, and two appendixes, and is structured as follows:

- In Chapter 1 "Wireless Sensor Network", we will deal with concepts of the Sensor nodes and both their software and hardware components, Wireless Sensor Network topologies, protocols, and related applications in real life.
- In Chapter 2 "Intrusion Detection Systems", we'll explain intrusions and their types. Next, we'll describe the IDS

methods of detections while focusing on anomaly-based IDSs. We'll also discuss the related work.

- In Chapter 3 "Our Approach : Machine learning-based IDS for WSN", we'll explain our approach, the tools we used, experiments and comment results.
- The codes and scripts related to the simulations and extraction of features are reported in Appendixes A and B respectively.

Chapter 1

Wireless Sensor Network

In this chapter, we describe the *Wireless Sensor Network* and its components, its topologies, protocols, and related applications in real life.

1 WSN definition

A wireless sensor network (WSN) is a collection of specialized self-sufficient devices (sensor nodes) with a communications infrastructure and a set of instruments that can measure, monitor, and record conditions at various locations [53].

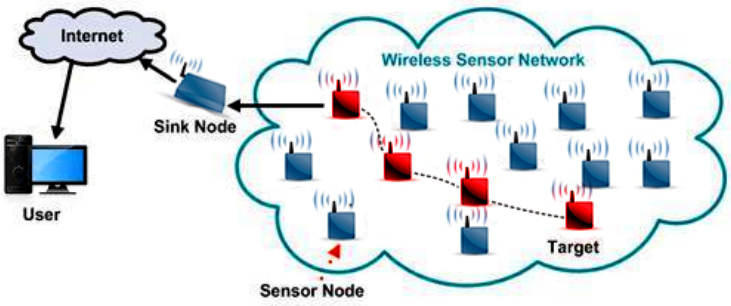

Figure 1.1: Wireless Sensor Network example

2 Sensor nodes

A Sensor node is an electronic component that is able to detect events or changes in its environment and processes this data than send the information[3].

the Sensor node can be seen as a set of Hardware and Software components.

2.1 Hardware Components

As shown in Figure 1.2 , a sensor node has several hardware components:

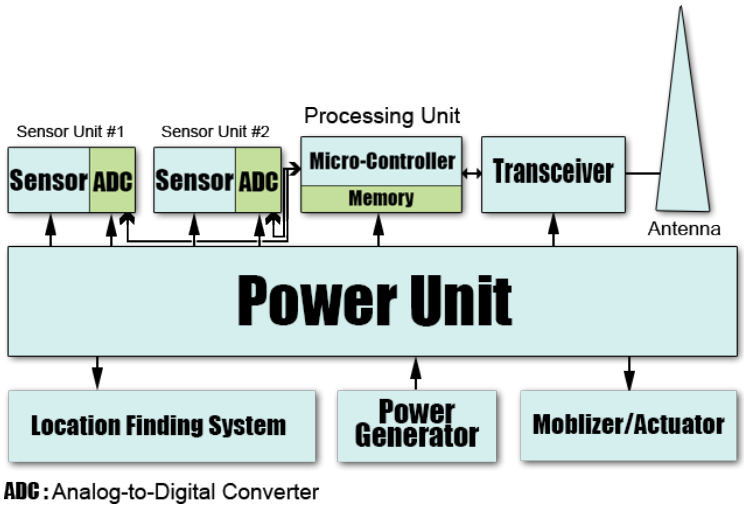

Figure 1.2: Sensor Node Components (Hardware)

- **Micro-controller** Micro-controller principal task is to process information that is either sensed locally or received by other nodes. Due to the nature of sensor nodes, these Micro-controllers have low-power design techniques such as sleep modes and dynamic voltage scaling, plus a limited computational power. Therefore the sensor node run component-based embedded operating systems (e.g. TinyOS) [22].
- **Memory** Sensor nodes maintain memory for the instruction set of the micro-controller and another memory for storing data and other local information (e.g.the node location). Memory is often limited in size due to economic reasons.
- **Radio transceiver** the radio transceiver included in Sensor nodes are low-rate, short-range wireless radio. Radio

communications are often the most power consuming operation, and hence the radio transceiver must incorporate energy-efficient sleep and wake-up modes.

- **Sensors with ADC unit** Due to the restrictions of energy, sensor nodes maintain low-data-rate sensing. Every node usually has several sensors implemented with an Analog-to-Digital Converter Unit (ADC), which translates the analog signals, that are provided by the sensors, to digital signals, which the controller will process afterward.
- **Location finding system** In many WSN applications, such as military operations, finding the location is necessary for analyzing the measured data. Unfortunately, only a few designers allow pre-configuring the sensor node location. Expressly for random WSNs and therefore satellite-based GPS have to be implemented [32].
- **Power Source** There are some WSN applications, in which the sensor nodes have energy harvesting techniques (like solar energy) which can accommodate a short amount of renewed energy or remain wired to a continuous power source. However, other applications continue to be stranded with the limited battery power, which most likely will be the bottleneck of this applications [38].

2.2 Software components

As observed in Figure 1.3 , a sensor node has a few software components.

- **Operating System Microcode** There are several Operating Systems for Sensor nodes such as TinyOS, Contiki, Mantis OS, and SOS. These operating systems are used for

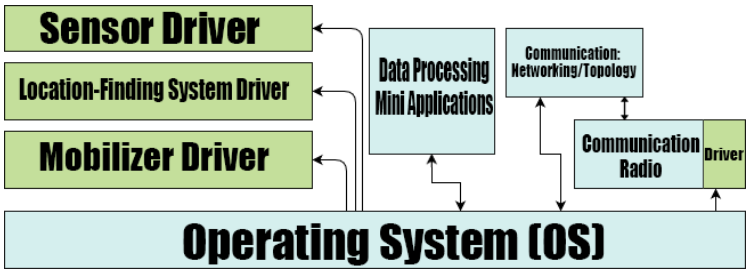

Figure 1.3: Sensor Node Components (Software)

general purposes from the machine-level functionality of the micro-controller to the application-level. The sensor node operating systems commonly consist of a rich collection software components and drivers [27].

- **Sensor Drivers** A sensor driver is a piece of code that manages the fundamental functions of sensor transceivers. It also includes the settings and configurations of this transceivers, depending on the sensor type.
- **Communication Processors** They handle all the communication actions such as routing, packets sending, buffering and forwarding, MAC layer contention mechanisms, spread-spectrum mechanisms, and the encryption of messages. They also sustain the topology maintenance [23].
- **Communication Drivers** These drivers are tasked with handling radio communication details, covering synchronization and clocking, encoding signals, recovering bits, byte counting, modulation, and signal levels [6].
- **Data Processing Mini-Applications** are the essential applications for processing the sensed data.

3 Topology

The topology of WSNs can vary from basic to advance and complex. The WSN most used topologies are Tree, Star, and Mesh.

3.1 Tree

The Tree topology uses a central hub called a root node as the main communication router. In the hierarchy, each mid-node of the tree is a central hub while the leafs are Sensor nodes as shown in figure 1.4. the network path in this topology can be single hop or multi hop.

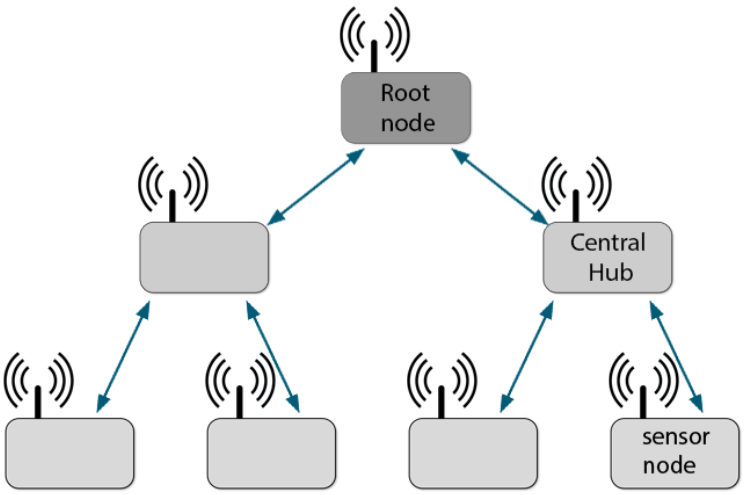

Figure 1.4: WSN tree Topology

3.2 Star

The nodes cannot communicate directly with each other and use a centralized communication hub (sink) instead. Therefore each node is a "Client" while the sink is the "Server" as shown in

Figure 1.5. Star networks have the disadvantage of single path communication [4].

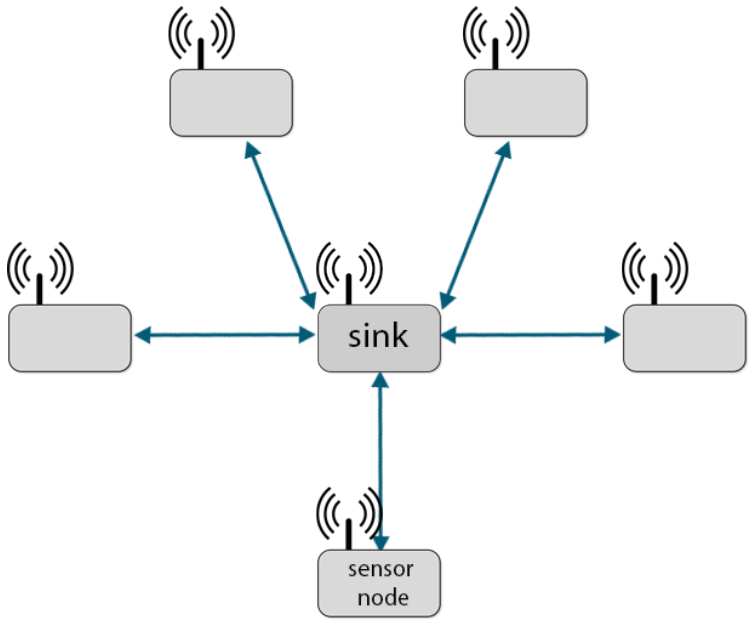

Figure 1.5: WSN Star Topology

3.3 Mesh

In the mesh topology, the nodes can have more than one adjacent, if every node is connected to all the other nodes, it's called Full Mesh. Else it's called Partial Mesh. The routing protocol determines the path for data transmission [4]. Mesh topology can provide an extension of network coverage without increasing transmit power or receive sensitivity, better reliability via route redundancy, easier configuration, and better battery life [50].

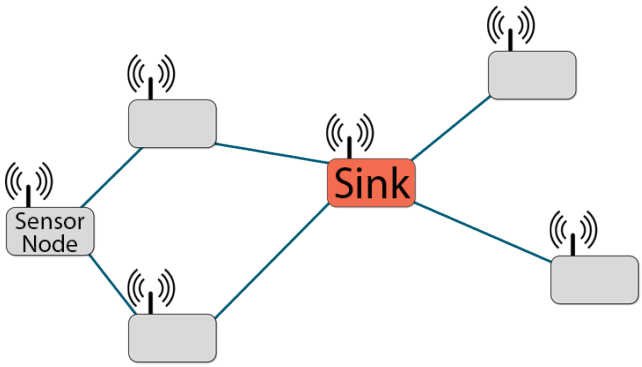

Figure 1.6: WSN Mesh Topology.

4 Routing protocols

The sensor nodes are battery-powered that are not changeable. Since the communications use a lot of energy, it must be kept to a minimum.Thus, Energy-Efficient Routing is one of the most important aspects of increasing the life span of sensors [40]. In what follows, we shall recall the strategy that any protocol has to respect.

4.1 Number of transmissions

Because every transmission uses energy, the strategy must lower the number of transmissions as much as possible, keep retransmissions to a minimum, and allow data to reach the sink with the least amount of hops.

4.2 Balancing activity between nodes

The protocol strategy must balance work between the nodes, so it won't be nodes that forward packets more than others and causing them to drain too much battery power. Which would

drive them to be either be dead or in battery save mode, leading to impossible communications.

4.3 Delay

For many applications, it is necessary that the delay of the transmission is not too big. It is advisable that the protocol reaches a compromise between overhead and delay [45].

4.4 Balancing the previous aspects

The routing strategy has to be aware of the energy resources that are left and has to balance all the previous aspects to produce the best possible solution.

4.5 Protocols categories

There are three principal categories for WSN protocols:

4.5.1 Pro-active Protocols

In proactive routing protocols, each node sends routing information regularly to every node in the network. These informations are saved in Routing Tables same way as wired networks. Which means that each node has an up-to-date Map of the network with all known routes. Proactive protocols have a smaller delay because routes are already established on request. They also produce higher overhead due to the constant exchange of tables. The most used Proactive protocols are Optimized Link State Routing (OLSR), Open Shortest Path First (OSPF), and Topology Broadcast based on Reverse-Path Forwarding (TBRPF) [42].

4.5.2 Re-active Protocols

Reactive routing protocols such as Associativity-Based Routing (ABR), Dynamic Source Routing (DSR), and Ad Hoc On-Demand Distance Vector Routing Protocol (AODV) are referred to as on-demand routing protocols. Different from proactive protocols, reactive protocols launch route discoveries only at when a node generates a request. If a route to a destination is required, a search procedure will be started to find a path from the requesting node to the destination. These protocols offer a lower overhead but produce longer delays [16]. In this work, we plan on using AODV protocol.

AODV: Ad Hoc On-Demand Distance Vector like all reactive protocols operates through allowing nodes to transmit topology information only on-demand. When a node wishes to send traffic to a host to which it has no route, it will produce a route request(RREQ) message and broadcast it to other nodes. When the RREQ message reaches either the destination itself or an intermediate node with a valid route entry for the destination, this node will generate and send a route reply (RREP). When the source node receives the RREP it begins to forward data packets to the destination [9].

4.5.3 Hybrid Protocols

Hybrid Protocols are a mixture of both, re-active and proactive routing. They aim to combine the advantages of both of these methods, by locally using pro-active and inter-locally using re-active routing. The idea behind hybrid routing is based on the fact that most communication in WSNs happens between nodes that are areal not wide apart and that changes in topology only

affect nodes in the neighborhood of the change. One approach to implement hybrid routing is to divide the WSN into several zones, and use pro-active routing inside each zone, and re-active routing between them. This protocol is called Zone Routing Protocol (ZRP) [52].

5 Applications

WSN have a wide range of real-life applications. Some of the application areas are environmental, military, health, and home. In the environmental use, for example, there are forest fire detection, seismic monitoring, flood detection and ecological habitat monitoring. In the military, sensor nodes ability to self-organize, deployed rapidly, and fault tolerance makes them very useful for battlefield surveillance, monitoring equipment, tracking targets, detecting nuclear and biological attacks. While in health, sensor nodes can also be deployed to monitor patients and assist disabled patients. Some other home applications include fire prevention, home automation, and security. [30].

6 Conclusion

In this chapter, we have recalled some basic concepts of wireless sensor network, the hardware, and software of sensor nodes. In addition, we have described their different topologies and protocols. Finally, the numerous real life applications of WSN are presented. In the next chapter, we'll review Intrusion detection systems the aim field of our study.

Chapter 2

Intrusion Detection Systems

In this chapter, we define the general meaning of an intrusion and its types. Then, we'll describe the IDS and its Analysis Engines that allow it to distinguish between different behaviors. We'll also address IDS methods of detections while focusing on anomaly-based IDSs. Finally, we'll discuss a couple of studies that approach a similar problematic to ours.

1 Intrusions

Intrusion is a set of action that attempts to jeopardize the integrity, confidentiality, or availability of any resource on the network [39].

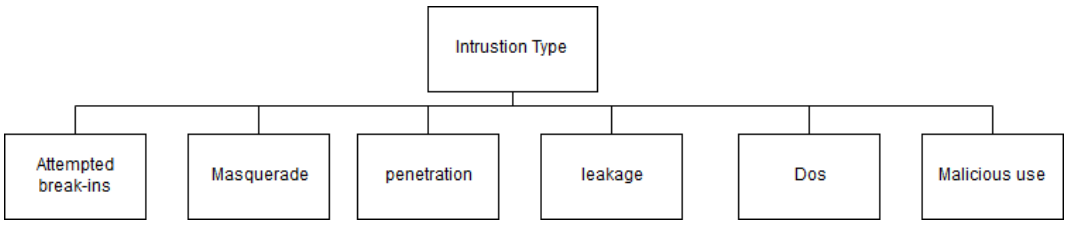

Figure 2.1: Intrusion types

Figure 2.1 shows a broad variety of intrusion attack categories [14], such as:

- **Attempted break-in:** Attempted break-in is an effort to have unauthorized access to the network, one of the attacks that fall under this category is "Brute Force." In which the attacker seeks to gain unauthorized access to a server by connecting to it and running a command, that tries multiple logins per second using a dictionary file of common passwords, while trying different combinations [19].

- **Masquerade:** is an attack in which the attacker tries to get a fake identity by using stolen passwords and logins, locating gaps in programs or finding a way around the authentication process. This identity would help him gain access to a personal informations, through legitimate access identification. This attack can be triggered either by someone within the organization or outside it (if it is connected to a public network) [8].
- **Penetration:** The possessioning of unauthorized access to the network. For example Pen-test: Penetration testing (also known as pen testing) is the practice of testing the WSN for vulnerabilities and system weaknesses that the attacker could exploit. [61]
- **Leakage:** An undesirable information flow from the network.
- **DoS:** In a denial-of-service (DoS) attack, an attacker attempts to prevent legitimate users from accessing information or services. The most common and obvious type of DoS attack occurs when an attacker "floods" a network with information. The sink can only process a certain number of requests at once, so if an attacker overloads the sink with requests. This is a "denial of service" because the sink becomes unresponsive [1, 25]. There are several attacks in WSN that fall under this category such as Blackhole, Wormhole, Hello Flood, Sybil Attack, Selective forwarding, and Exhaustion attack [37, 44].
- **Malicious use:** Deliberately harming the network resources.

2 Intrusion Detection Systems

An Intrusion Detection System (IDS) is an incorporation of hardware and software, which examines system or network activity to detect possible malicious behaviors. In a case of discovery, the IDS alerts the system or the network administrator [12]. The principal objective of an IDS is to be able to distinguish different attacks and normal behavior (high detection rate and a low false alarm rate) [47]. Network-Based Intrusion Detection Systems (NIDS) are the most popular, they monitor the network, capture and examine all the packets that are flowing over. They can also examine the entire packet, not just IP addresses and ports but look for the payload within a packet, to recognize the accessed particular host application, and with what configurations, then, raise alarms when an attacker exploits a bug in such code. By detecting known attack signatures. NIDS are independent to the host and examine passing network traffic for signs of intrusion [34]. Host-Based Intrusion Detection Systems (HIDS) monitor system records using log files and periodically compare to the previously gathered information, like the owner of these files, file size, last modification date and such. This process allows it to detect various intrusions [18, 41]. There are three main sets of analysis engines for IDS:

- **Event or Signature-based Analysis**
- **Statistical Analysis**
- **Adaptive Systems**

The Event or Signature-based Systems functions similarly to the anti-virus software which most people are familiar with. It produces a list of patterns that are deemed to be suspicious or indicative of an attack. The IDS merely scans the environment

looking for a match to the known patterns. The IDS then responds by taking a user-defined Action by sending an alert or performing additional logging. This is the most common type of intrusion detection systems; a *Statistical Analysis System* which builds statistical models of the environment, such as the average length of a connection session, then looks for deviations from "normal". There is a considerable amount of active research in this area, and some producers are just beginning to incorporate this technology in the form of marketable products. *The Adaptive Systems* start with generalized rules for the environment, then learn, or adapt to, local conditions that would otherwise be unusual. After the initial learning period, the system understands how people interact with the environment, and then warn operators about any unusual activities. It's important to mention that any IDS will both miss some kinds of suspicious activity (false negatives) and signal alarms when there is nothing wrong (false positives). This is why organizations must have a strong human process that interacts with the IDS to evaluate the operating environment. The machine intelligence of most intrusion detection systems is still evolving and current research is both following step-by-step this evolvement and actively working to facilitate and accelerate it[2].

Figure 2.2 shows placement of a traditional network-based IDS with two sensors on separate network segments that communicate with a monitoring station on the local network.

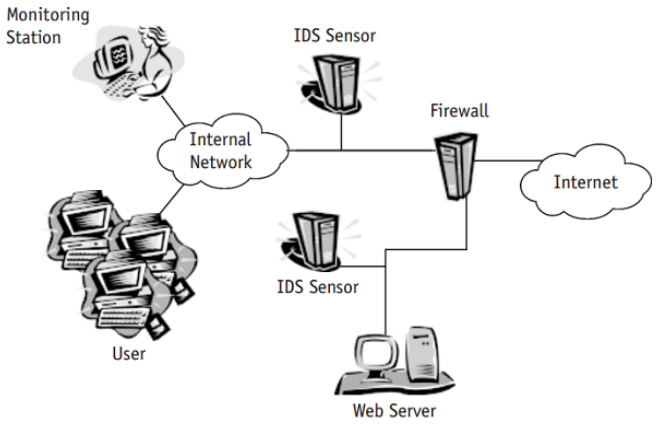

Figure 2.2: Traditional network-based

IDSs are functionally categorized into three groups: anomaly-based detection, misuse-based detection, and specification-based detection (see Figure 2.3).

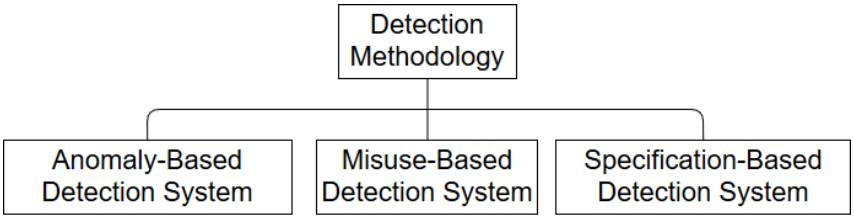

Figure 2.3: Detection Methodologies

2.1 Anomaly-Based Detection System

Anomaly-Based Intrusion Detection System (ABIDS) has brought numerous researchers due to its ability to recognize novel attacks (unknown attacks). Novel attack detection is a method for recognizing an unknown attack, even though, the machine learning system is not aware of during training [43]. ABIDS has two principal advantages over Specification-Based Intrusion Detection System (SBIDS). The first advantage is

the ability to detect unknown and "zero day" attack. This is done by comparing the normal behavior and what deviate from it. The second advantage is the customization of the normal behavior profile for the system and network. This customization makes it puzzling for an attacker to perceive with certainty what activities can he carry out without being detected [7]. The response of the system depends on the way of implementing and testing all protocols. The major shortcoming of anomaly detection is defining its rule set [31]. Anomaly-based detection system has several detection techniques: Statistical-based, Data Mining-based, Knowledge-based, and Machine learning-based detection. The complete classification of ABIDS illustrated in Figure 2.4.

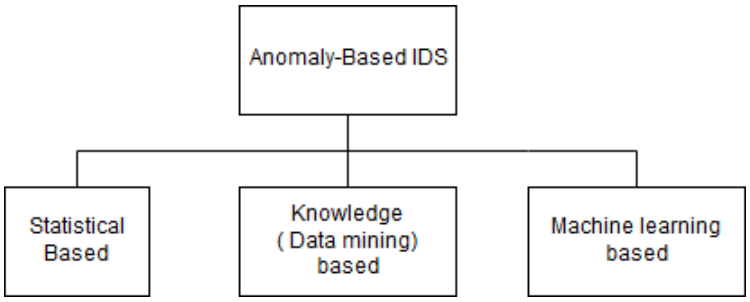

Figure 2.4: Classification of Anomaly-Based techniques

2.1.1 Statistical-Based Detection Technique

In electronic systems, Statistical modeling is one of the latest techniques used for detection intrusions. statistical properties and statistical test are used by Statistical-based anomaly detection techniques to determine whether the observed behavior strays considerably from anticipated behavior[54].

2.1.2 Data Mining-Based Detection Technique

Although it is unable to detect insider attack, IDS can essentially only detect known attacks on their signatures. Data mining, the process of extracting useful and previously ignored models or patterns from large stored data, can be regarded as a better solution for IDS at its best to "pattern finding". The data mining process is likely to decrease the amount of data that must remain stored for historical comparison of network activity, thus, creating data that are more useful for the detection of anomalies. [20, 49].

2.1.3 Knowledge-Based Detection Technique

The knowledge-based detection technique collects knowledge about the specific attacks and system weaknesses and then applies this knowledge to exploit the attack and weaknesses to produce the alarm [14].

2.1.4 Machine Learning-Based Detection Technique

Machine learning is a technique which can be described as the ability of a program or a system to learn and improve their performance for certain tasks or a group of tasks over time. It primarily concentrates on establishing a system for improving the performance on the basis of the previous result. It is used in IDSs for its capability of recognizing attacks. Machine learning has the ability to alter the execution strategy-based on newly acquired information [58].

2.2 Misuse-Based Detection System

Intrusions are discovered through actual matching behavior recorded in audit trails with known suspicious patterns. Although misuse detection is entirely effective in discovering known attacks, it is useless against novel attacks for which the signatures are not yet available. Furthermore, it is challenging to build a signature that incorporates all possible variations of known attacks. Any mistakes in the definition of these signatures will decrease the effectiveness of this detection technique and increase false alarm rate [24].

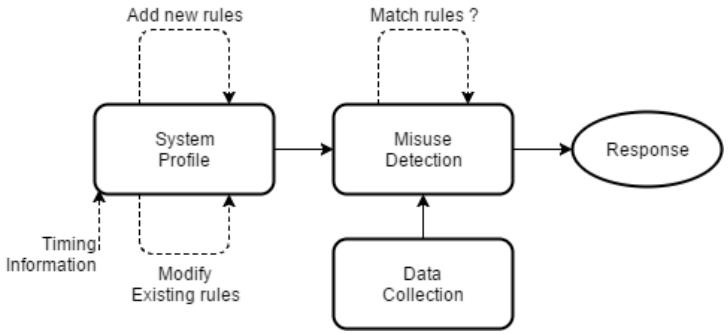

Figure 2.5: Misuse Detection System

Figure 2.5 illustrates the principle functionality of Misuse Detection System. The latter consists of four parts: data collection, system profile, misuse detection, and response. Data are obtained from one or many data sources such as audit trails, network traffic, system call trace, etc. Collected data are transmitted in a format that is comprehensible by the other parts of the system. The system profile is used to distinguish between normal and malicious behaviors. The profiles characterize normal behavior (what it should be and what operations it would typically perform or not perform). The profiles are constantly matched with actual system activities and reported as intrusions

in case of aberrations.

2.3 Specification-Based Detection System

Specification-based intrusion detection techniques are able to take the best of the misuse and anomaly-based detection techniques. To do so, manually developed specifications and constraints to characterize legitimate system behavior are used. Like anomaly-based detection techniques, Specification-based intrusion detection techniques can detect attacks as the aberrations from a normal behavior. Specification-based detection techniques have low false alarm rate compared to the high false alarm rated anomaly-based detection techniques. The reason behind having low false alarm rate is that they are-based on manually detection techniques and manually developed specifications and constraints. However, time would be the cost to achieve such a low false alarm rate for the development of detailed specifications and constraints are time-consuming [57].

3 Related Work

Anomaly-Based Detection System needs a set of features to describe the different behaviors of a sensor node, to set apart which can be considered Normal from which is Malicious. In our work, we are interested in studies that covers the impacts of attacks on WSNs behavior. We will make a summary of the two most related to our work.

Darra, Skouloudi and Katsikas [17] present a simulation analysis for DoS attacks against WSNs. They simulate and analyze the performance of routing protocols for WSNs using a scenario-based experiment, to analyze the network's behavior

under all the simulated attacks. Namely Blackhole, Flooding, Rushing and Selective Forwarding Attacks. The authors simulated several attacks by observing several network features. The objective of this study is to detect what are the most revealing parameters toward developing suitable IDS.

Dini and Tiloca [21] present a simulative approach to analyzing attack impacts. They show that simulation results provide valuable insights on the attack severity. They achieved that by three steps. First, they evaluate the effects of some attacks on networks. Next, they quantify attack impact with a set of security metrics. They classify different attacks according to their severity. Finally, they evaluate different security solutions against considered attacks, Which enables a designer to compare their capability and choose the most suitable ones.

The two above studies aim to have more ideas on features that discriminate behaviors in WSNs. In our work, we rely on classifiers to characterizes different attacks in WSNs.

4 Conclusion

In this chapter, we have given an overview of various Intrusions, The different detection methodologies of IDS are described with a review of related work discussion. In the next chapter, we'll discuss our approach, Experiments, and results.

Chapter 3

Our Approach : Machine learning-based IDS for WSN.

In this chapter, we describe the system that we have designed to fix the well practical features needed to compose an IDS for WSN. which is based on machine learning techniques and Simulation data. We will justify all done choices about the level of attack characterization, the targeted features, and the used classifiers. Then, we describe our experimental study and its given results.

1 Overview of our System

The main idea is to build a machine learning system to detect attacks while varying several features to get the most efficient ones.

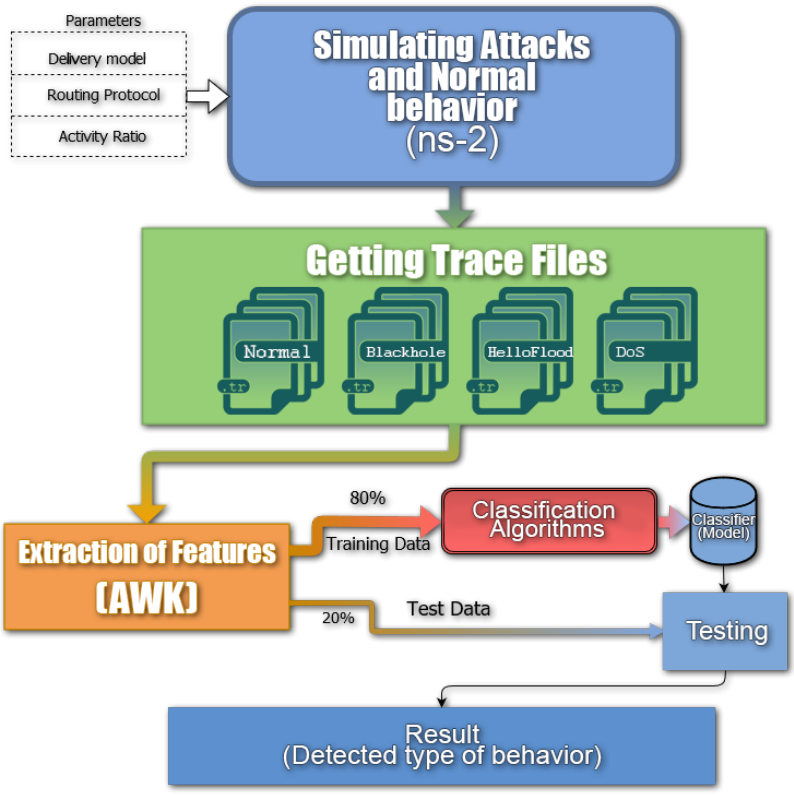

Figure 3.1: Global view of the system

Our study relies on simulations. Fig 3.1 illustrates the proposed system. First, we launch a simulation of normal and malicious behaviors under many scenarios; The outputs are a batch of trace files that capture our simulations. From these files, we extract various features that define those behaviors (normal and malicious) using several classifiers we generate Normal and Malicious models. In what follows we detail each of this steps.

1.1 Simulation

Concerning the data delivery model, there are several to consider depending on the application nature. In fact, the data delivery model to the Sink can be Continuous, Event-driven or Query-driven or Hybrid [55]. In the continuous delivery model, each sensor sends data periodically, while Event-driven and Query-driven models, the transmission of data initiates when an event occurs, or when the sink generates a query. In our experiment, and to be more general, we have picked event-driven data delivery model in which a node send data to the sink.In fact, we model the time of events occurring randomly. In addition to Normal behavior, we adopt three types of attacks that mainly fall under Denial of Service category: Blackhole, Hello Flood, and DoS we have chosen these attacks as they are the most frequent and the most harmful.

Moreover, To have numerous scenarios we have considered another parameter which fixes the number of the working nodes (Activity Ratio parameter). This parameter is added to measure the performance of an IDS under many workload scenarios. For the sake of simplicity, we have opted for AODV protocol.

1.2 Classification Algorithms

In order to build classifiers for the different attacks, we need to adjust some concerns, that fix the:

1. Level of characterization,
2. Features to extract,
3. Algorithms of classifications.

1.2.1 Which Level ?

To characterize an attack in WSN, we need to determine at which level the characterization should be conducted. One can do that at the:

- **Node level:** monitoring each node activity.
- **K-hop level:** In this level, we characterize all k-hop communications between nodes.
- **Connection level:** Capturing each communication between the sending node to the sink.

In this study, we have investigated the last level, in order to deal with less amount of data.

1.2.2 Which Features ?

Many features can be considered to characterize a behavior at the connection level. In our study, we have examined some of them as shown in Table 3.1. We have classified these features according to their type : Data, Routing, and Time.

Feature	Type	Description
Duration	Time	The amount of time between sending the first packet and receiving the last packet.
All Received Packets	Data, Routing	All the packets (regardless of who sent them and their type) that the sink received during the duration of this connection
AODV Route Request	Routing	The number of AODV RREQ packets that the sending node has sent.
AODV Route Reply	Routing	The number of AODV RREP packets that the sending node has sent.
TCP packet sent	Data	The number of TCP packets which the sending node has sent.
TCP packet received	Data	The number of TCP packets that the sink has received.
TCP packet dropped	Data	The number of TCP packets that got dropped.
TCP packet forward	Data	The sum of the number of times that each TCP packet in this connection got forwarded.
Energy consumed	-	The amount of energy consumed by all the nodes inherent to the connection.
Packet delivery ratio	Data	the ratio of received TCP packets to sent TCP packets.
Average delay	Time	The average delay of the received TCP packets during the connection.
Max hop	Routing	Maximum number of hops during the connection.
Average hop	Routing	Average number of hops during the connection.
Throughput	Data	the amount of received data in the connection duration (kbps).

Table 3.1: Descriptions of the targeted Features

1.2.3 Which Classifier ?

Several classification algorithms can be applied. In our study, we have considered four of these algorithms which we have chosen according to their popularity.

- **Random Forests** or Random Decision Forests are a set of learning techniques for classification. They operate by assembling a multitude of decision trees at training time and outputting the class that is the mode of the classes

(classification) of the individual trees [28].

- **SVM** Support Vector Machine (SVM) is a discriminative classifier formally described by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples [48].
- **Decision Table** are classification models used for prediction. They consist of a hierarchical table in which every entry inside a higher level table gets broken down by the values of a couple of additional attributes to form different table. The formation is equivalent to dimensional stacking. A visualization method is manifested, which allows a model based on many attributes, to be recognized even by those unfamiliar with machine learning [11].
- **IBK** Instance based k-Nearest-Neighbor (kNN) classification algorithm is a non-parametric method used for pattern recognition [5]. It depends on having a correctly labeled dataset of inputs. kNN locates the nearest k input vectors in its dataset to the input query. Then, it joins the labels of these k closest points from the dataset to predict the correct output value [60].

2 Experiments and Results

Let us recall that the primary objective of this work is to determine which features are the best to characterize attacks. We have performed a batch of experiments that aims to:

– Pick the suitable classifier among Random Forest, SVM, IBK and Decision Table.

– Accumulate the sub-optimal set of features that are sufficient to perform on an IDS with the best possible quality and lower complexity.

First, we specify the application model of our WSN. Then, we describe the used tools in this experiment. Next, we detail the process of simulating both normal and malicious behaviors under ns-2. Finally, we present the results and their interpretation.

2.1 Application Model

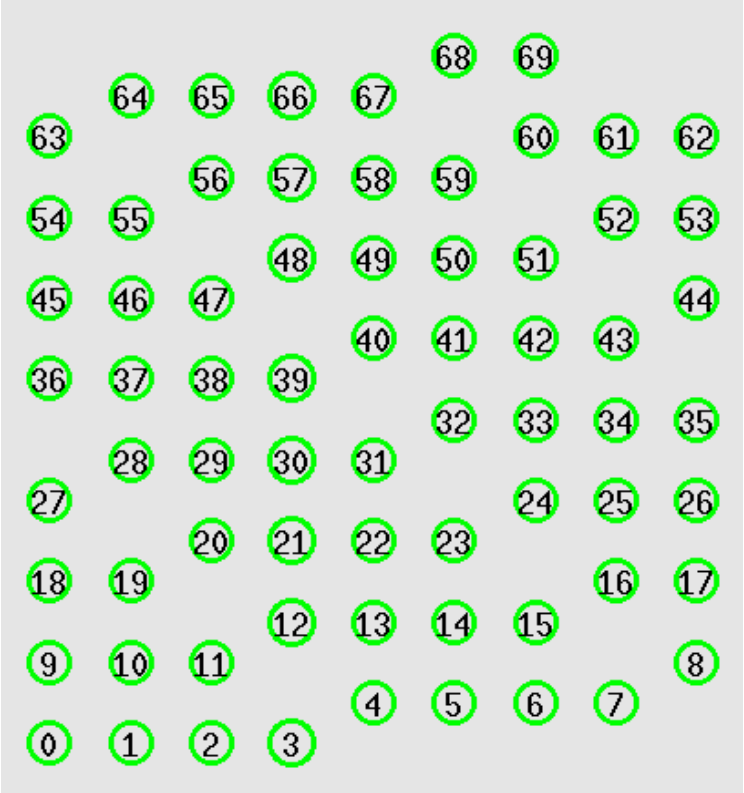

Figure 3.2: Application model

We have considered a simple environment, where sensor nodes are placed in mesh topology as shown in Figure 3.2. The node with the identification "0" is regarded as the Sink. Without loss

of generality, we have used 70 nodes in our simulations. This choice is made to ease visualization.

Let us recall that the data delivery model is event-driven. The nodes sense values such as temperature, pressure, sound, etc. when these values reach a certain threshold, it gets transmitted to the sink.

Concerning the routing, we performed our analysis using AODV. Our choice fell on this protocol due to its adequacy with WSN. in addition to the ability to support mobility.

More detail, on ns-2 configurations are presented in Appendix A from Section 1 to Section 4.

2.2 Used Tools

In order to make our results reproducible, we have relied on a set of Open Source Tools and languages.

2.2.1 ns-2

ns-2 is an Open-Source simulation tool, which runs on Linux. It is a discrete event simulator targeted at networking research and grants a solid support for simulation of routing, multicast protocols and IP protocols, such as UDP, TCP, RTP, and SRM across wired and wireless networks. It also has support for multiple protocols and the capability of graphically detailing network traffic. Furthermore, ns-2 carries sparse of algorithms in routing and queuing. Routing algorithms such as AODV, DSR, and DSDV. Queuing algorithms include fair queuing, deficit round-robin and FIFO [33].

2.2.2 GAWK

GNU AWK is an Open-Source implementation of the AWK programming language, and it is available for all UNIX® systems. The AWK language is a data-driven scripting language, powerful text manipulation tool, and pattern matching language that is especially useful for data extraction [56]. As shown in Figure 3.3 AWK follows a simple workflow, First, executing the Begin{} block, then, reading the input a line at a time, while executing the Action{} block on these lines, after completing the input, it executes the End{} block.

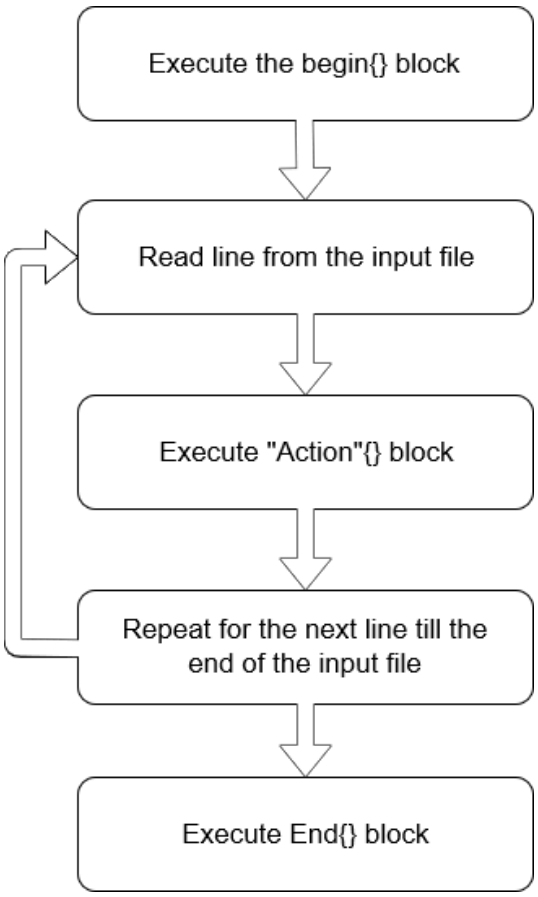

Figure 3.3: AWK script workflow

2.2.3 WEKA

The Waikato Environment for Knowledge Analysis is a Java software, with graphical user interfaces that make it easier to access a set of visualization tools, data analysis algorithms and state-of-the-art techniques in machine learning [62].

2.3 Behaviors Simulation and Parameters

We have simulated the Normal and three different malicious behaviors: Blackhole, Hello Flood, and DoS.

Concerning the Normal behavior, we have chosen AODV protocol for routing and TCP for transmission. There are 70 nodes in our topology, while the delivery plan is at a given time, a node sends data to the sink (node 0). For the sake of diversifying scenarios, we have varied the number of working nodes in a scenario. This is achieved by considering four different activity ratios (10%, 25%, 50%, 75%) in each scenario. In order to simulate the event-driven delivery model, the node starts sending data at random time, doing so, we have simulated real life applications. The ns-2 TCL script of the Normal behavior is presented in Appendix A Section 8.

For reproducing the same Normal scenario when simulating attacks, we snapshot the scenario (Saving the identity of working nodes and their data transmission starting time). The ns-2 TCL script to perform that is detailed in Appendix A Section 5.

Regarding the malicious behaviors, we generated the same topology as Normal behavior with the same parameters, then we load the file that contains the nodes identities, and their sending data starting times. Next, we picked randomly one single node to act maliciously. We chose three different malicious behaviors :

- **Blackhole** The malicious node drops all the packets passing through it. In order to do this, it attracts its neighbor nodes by forging route reply with less hop count and greater sequence number. We have simulated that by adding this malicious act to the AODV protocol as reported in Appendix A Section 9.1 and the TCL script in Section 9.2 .
- **Hello Flood** The flooder node keeps sending RREQ messages despite receiving RREP messages, in an attempt to waste the network bandwidth and exhaust its resources. We simulated this by adding the code shown in Appendix A Section 10.1 to AODV protocol and TCL script is detailed in Section 10.2.
- **DoS** The attacking node keeps on sending packets to the sink, in an attempt to make it unresponsive. the TCL script is presented in Appendix A Section 11.

Following the previously described scenarios, we have launched the simulations. That led to a set of trace files from which we extracted the targeted features. We have deployed AWK scripting to perform this extraction. The script is reported in Appendix B. Each connection is characterized by all features.

Activity Ratio	Total Number of Scenarios	Number Of Connections
10%	3,816	23,393
25%	1,484	23,533
50%	764	23,632
75%	504	23,868
Total	6,568	94,426

Table 3.2: Statistics on the dataset.

Figure 3.2 reports the numbers of made scenarios and their deduced connections. We divided this dataset into 80% for

training and 20% for testing. For the generation of models and the measuring of performances, we have used WEKA tool.

2.4 Results

With the purpose of measuring the performances of our Machine Learning-based IDS, we have first compared the chosen classifiers performances. Figure 3.4 illustrates comparative results between Random Forest, SVM, Decision table, and IBK Classifiers in terms of Recall and Precision.

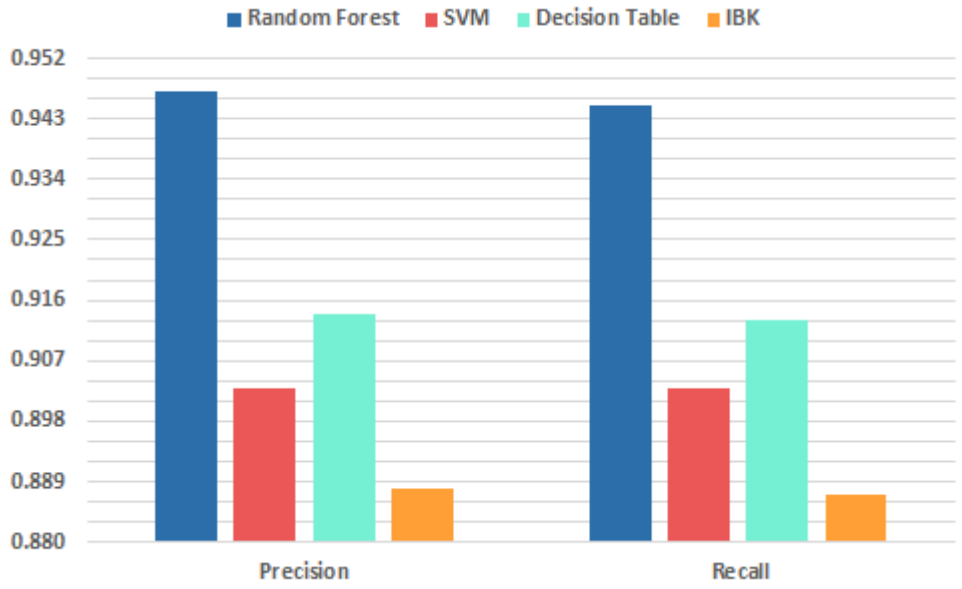

Figure 3.4: Performances of Classification of different Classifiers

For the sake of examining these algorithms, we selected all the Features for classification. As shown in Figure 3.4, the Random Forest classifier is the most suited for distinguishing behaviors.

Now, relying on Random Forest classifiers, we attempt to get the most suitable sub-optimal set of features. We have chosen three set of features.

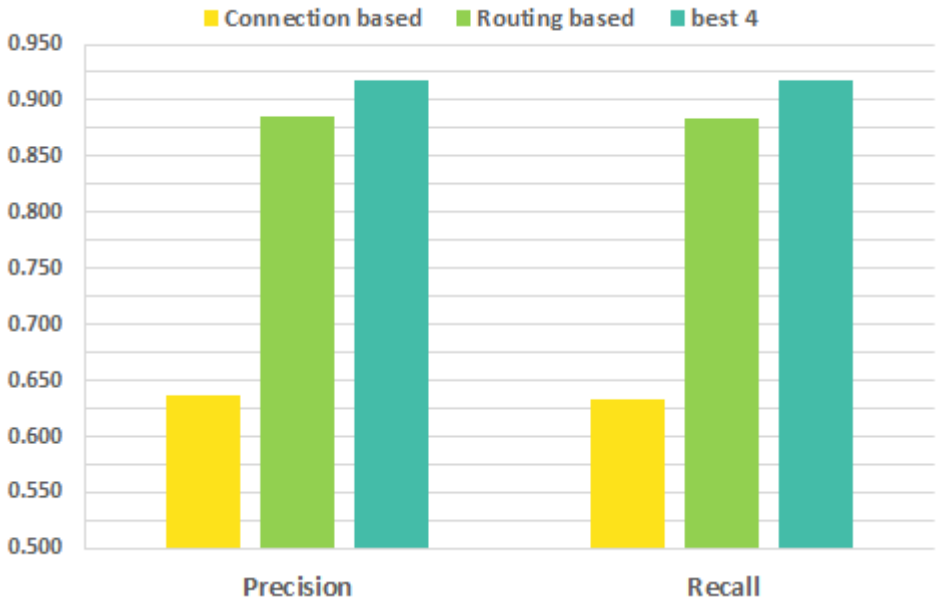

Figure 3.5: Performances according to Features

For the first set, we only picked the features related to the connection (Duration, TCP sent packets, TCP received packets, TCP forwarded packets, Average delay, and Throughput). In the second set, only the features more related to routing are selected (AODV Route Request, AODV Route Reply, Energy consumed, and Average hop). While in the third set, we have deployed a Machine Learning technique which decides the most influential Features. In fact, we got the four best attributes (AODV Route Request, Packet Delivery Ratio, Average delay, and Throughput). For that we have used Gain Ratio Attribute Evaluation Algorithm, which determines the value of an attribute, by estimating the gain ratio while respecting the class [10]. Figure 3.5 report comparative results of classification using these three sets. The performances are reported in terms of Precision and Recall. We observe that the *Routing-based* Features and *Best4* deliver better results than *Connection-based* Features.

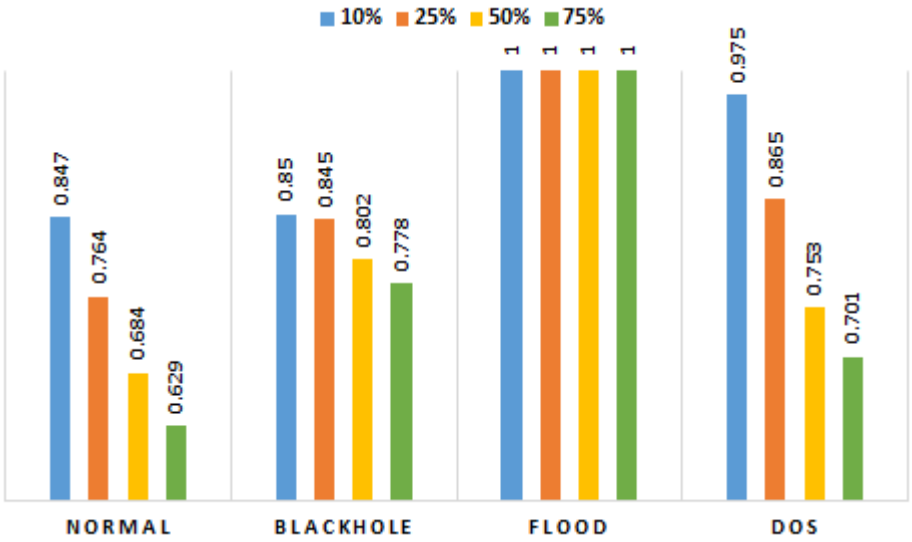

Figure 3.6: Performances classification regarding the activity ratio

Figure 3.6 reports the comparative performances of the classification in term of F-Measure according to the activity ratio. The first observation is that Hello Flood attack is the easiest one to be detected over all the WSN workload scenarios. These results also prove that the more workload in WSN is, the less attacks are detectable, especially for DoS attack. In fact, we observe a deviation of more than 27% between F-measure of 10% activity ratio and 75% one.

For instance, we examine the matrices of Confusion in 75% Activity Ratio Scenario in Table 3.3.

	a	b	c	d
a	1052	134	0	23
b	194	953	0	6
c	0	0	1159	0
d	29	3	0	1126

a) 10%

	a	b	c	d
a	936	130	0	155
b	175	943	0	21
c	0	0	1209	0
d	117	19	0	1002

b) 25%

	a	b	c	d
a	936	130	0	155
b	175	943	0	21
c	0	0	1209	0
d	117	19	0	1002

c) 50%

	a	b	c	d
a	734	166	0	289
b	174	919	0	99
c	0	0	1243	0
d	236	84	0	830

d) 75%

Table 3.3: Confusion Matrices

From the Matrix of confusion, We observe that the DoS attack is confused with the Normal behavior.

3 Conclusion

In this chapter, we have presented our system which we have designed to determine the sub-optimal set of features needed to compose a Machine learning-based IDS for WSN. The dataset is the result of preprocessing simulation data. We have justified all made choices on the level of attack characterization, the targeted features, and the used classifiers. We described our experimental study and commented the given results.

Conclusion and Future Work

In this work, we have dealt with Wireless Sensor Networks (WSNs) generalities and their applications, we have also mentioned how that is important to secure these networks using Intrusion Detection Systems (IDSs). The latter must deploy lightweight solutions that deliver high performance while respecting WSNs specificities and limitations in terms of energy, memory, and computation power.

Our primary goal was to find a sub-optimal set of features that can characterize attacks, and use this features for Machine Learning-Based IDS. In order to do so, we have launched numerous simulation scenarios that contain several targeted behaviors. These behaviors are Normal, Blackhole, Hello-Flood, and DoS. Next, we extracted features in the connection level from these simulations. Later, we compared several combinations of the targeted features and classifiers for the sake of finding the sub-optimal features and the right classifier.

Our results have shown that only four Features are needed for characterization using Random Forest Classifier.

In the future, more investigation can be done concerning node activity level and k-hop level. This work can be riched by trying other simulation configurations and protocols.

Appendices

Appendix A

NS2 simulated behaviors

1 Nodes configuration and settings

```
set val(chan)           Channel/WirelessChannel     ;# channel type
set val(prop)           Propagation/TwoRayGround    ;# radio-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)         100                         ;# max packet in ifq
set val(nn)             70                          ;# number of mobilenodes
set val(rp)             AODV                        ;# protocol tye
set val(x)              200                         ;# X dimension of topography
set val(y)              200                         ;# Y dimension of topography
set val(stop)           10                          ;# simulation period
```

2 Initializing objects and trace files

```
set ns                  [new Simulator]
set tracefd             [open Normal.tr w]
set namtrace            [open normal.nam w]

# set up a new trace file format
$ns use-newtrace
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

set dist(25m) 3.07645e-07

Phy/WirelessPhy set CSThresh_ $dist(25m)
Phy/WirelessPhy set RXThresh_ $dist(25m)
```

3 Setting topography and values to the configured parameters

```
# set up topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)

#creating General Operations Director
create-god $val(nn)

# configure the nodes
$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 [new $val(chan)] \
-topoInstance $topo \
-energyModel "EnergyModel" \
-initialEnergy 10 \
-txPower 0.33 \
-rxPower 0.1 \
-idlePower 0.05 \
-sleepPower 0.03 \
-agentTrace ON \
-routerTrace ON \
-macTrace  OFF \
-movementTrace OFF \
```

4 Creating and positioning nodes

```
for {set i 0} {$i < $val(nn) } { incr i } {
set mnode_($i) [$ns node]

$mnode_($i) set X_ [expr [expr [expr $i * 20] % 180] + 10]
$mnode_($i) set Y_ [expr [expr [expr $i * 70] / 280] * 10]
$mnode_($i) set Z_ 0.0
}

$mnode_(0) label "Sink"

for {set i 0} {$i < $val(nn)} { incr i } {
$ns initial_node_pos $mnode_($i) 10
}
```

5 Saving normal behavior scenario

```
for {set i 0} {$i < $wn} { incr i } {
set random_node [expr int ( rand() * 69 ) + 1 ]
#nodes file adding lines
puts $nodesfile "$random_node"
#Setup a TCP connection
set tcp_($i) [new Agent/TCP]
$tcp_($i) set class_ 2
$tcp_($i) set fid_ [expr $i + 1]
set sink_($i) [new Agent/TCPSink]

$ns attach-agent $mnode_($random_node) $tcp_($i)
$ns attach-agent $mnode_(0) $sink_($i)
$ns connect $tcp_($i) $sink_($i)

set ftp_($i) [new Application/FTP]
$ftp_($i) attach-agent $tcp_($i)
```

```
}

for {set i 0} {$i < $wn} { incr i } {
set start_time [expr ( rand() * 9)]
set stop_time [expr $start_time + 0.5]

puts $timefile "$start_time"

$ns at $start_time "$ftp_($i) start"
$ns at $stop_time "$ftp_($i) stop"
}
```

6 Loading normal behavior scenario

```
foreach line $nf {
#Setup a TCP connection
set tcp_($j) [new Agent/TCP]
$tcp_($j) set class_ 2
$tcp_($j) set fid_ [expr $j + 1]
set sink_($j) [new Agent/TCPSink]

$ns attach-agent $mnode_($line) $tcp_($j)
$ns attach-agent $mnode_(0) $sink_($j)
$ns connect $tcp_($j) $sink_($j)

if {$random_node == $line} {
set random_node [expr int ( rand() * 69 ) + 1 ]
}

set ftp_($j) [new Application/FTP]
$ftp_($j) attach-agent $tcp_($j)
incr j
}

set j 0
foreach line $tf {
set start_time $line
set stop_time [expr $start_time + 0.5]
$ns at $start_time "$ftp_($j) start"
$ns at $stop_time "$ftp_($j) stop"
incr j
}
```

7 Ending simulation and closing the used files

```
# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
$ns at $val(stop) "$mnode_($i) reset;"
}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at [expr $val(stop) + 0.01] "puts \"end simulation\"; $ns halt"
proc stop {} {
global ns tracefd namtrace
$ns flush-trace
close $timefile
close $nodesfile
close $tracefd
close $namtrace
}
```

8 Normal behavior

```
//Nodes configuration and settings

#setting nbr of the nodes that would send data
set wn [lindex $argv 0]
# opening Files on writing mode to save starting times and the sending nodes
set timefile [open timef.txt w]
set nodesfile [open nodes.txt w]

//Initializing objects and trace files
//Setting topography and values to the configured parameters
//Creating and positioning nodes
//Saving normal behavior scenario
//Ending simulation and closing the used files
```

9 Blackhole attack

9.1 the added code to AODV.h

```
//in AODV routing agent class
bool malicious;
//rest of the code is added to AODV.cc
```

9.2 TCL script

```
//Nodes configuration and settings

#setting nbr of the nodes that would send data
set wn [lindex $argv 0]
# opening Files on reading mode to load starting times and the sending nodes
set timefile [open timef.txt r]
set nodesfile [open nodes.txt r]
set tf [read $timefile]
set nf [read $nodesfile]
set j 0

//Initializing objects and trace files
//Setting topography and values to the configured parameters
//Creating and positioning nodes
//loading normal behavior scenario

set random_node [expr int ( rand() * 69 ) + 1 ]
$ns at 0.0 "[$mnode_($random_node) set ragent_] blackhole"
puts "$random_node"

//Ending simulation and closing the used files
```

10 Hello flood attack

10.1 the added code to AODV.h

```
#define FLOOD_INTERVAL  0.09
//in Timers
class FloodTimer : public Handler
{
public:
```

```
FloodTimer(AODV* a) : agent(a){}
void      handle(Event*);
private:
AODV      *agent;
Event      intr;
};
//in AODV routing agent class
friend class FloodTimer;

//in Packet TX Routines
void  FloodRREQ(nsaddr_t dst);
//in AODV:Agent Timers
bool  flooder;
FloodTimer ftimer;
//rest of the code is added to AODV.cc
```

10.2 TCL script

```
//Nodes configuration and settings

#setting nbr of the nodes that would send data
set wn [lindex $argv 0]
# opening Files on reading mode to load starting times and the sending nodes
set timefile [open timef.txt r]
set nodesfile [open nodes.txt r]
set tf [read $timefile]
set nf [read $nodesfile]
set j 0

//Initializing objects and trace files
//Setting topography and values to the configured parameters
//Creating and positioning nodes
//loading normal behavior scenario

set random_node [expr int ( rand() * 69 ) + 1 ]
$ns at 0.0 "[$mnode_($random_node) set ragent_] flooder"
puts "$random_node"

//Ending simulation and closing the used files
```

11 DoS attack

```
//Nodes configuration and settings

#setting nbr of the nodes that would send data
set wn [lindex $argv 0]
# opening Files on reading mode to load starting times and the sending nodes
set timefile [open timef.txt r]
set nodesfile [open nodes.txt r]
set tf [read $timefile]
set nf [read $nodesfile]
set j 0

//Initializing objects and trace files
//Setting topography and values to the configured parameters
//Creating and positioning nodes
//loading normal behavior scenario

#creating a DoS attack
set udp [new Agent/UDP]
$udp set fid_ $j
set null [new Agent/Null]
$ns attach-agent $mnode_($random_node) $udp
$ns attach-agent $mnode_(0) $null
```

```
set cbr [new Application/Traffic/CBR]
$cbr set interval_ 0.001
$cbr set random_ 1
$cbr set packetSize_ 1000
$cbr set maxpkts_ 10000

$cbr attach-agent $udp
$ns connect $udp $null

$ns at 0.0 "$cbr start"
$ns at 10.0 "$cbr stop"

//Ending simulation and closing the used files
```

Appendix B

AWK script

```
BEGIN {
# Initial Energy assigned to each node in Joules
flows = 70;
cnxs = 0;


start_time[flows] = 0.000000000;
end_time[flows] = 0.000000000;

#all_sent[flows]
pkt_delivery_ratio[flows] = 0;
total_hop_count[max_] = 0;
max_hop_count[flows] = 0;
avg_hop_count[flows] = 0;
overhead[flows] = 0;
start = 0.000000000;
end = 0.000000000;
packet_duration = 0.0000000000;
recvnum = 0;
delay[flows] = 0.000000000;
sum = 0.000000000;

i=0;
j=0;
total_energy_consumed[flows] = 0.000000;

aodv_rreq[flows] = 0;
aodv_rrep[flows] = 0;

sender_node[flows] = 0;
flow_begining_time[flows]=0.00000000000000;

all_packets[flows] = 0;
total_rcvd_byte[flows] = 0;


for(i=0;i<flows;i++){
start_time[i] = 10.000000000;
packet_drop[i] = 0;
packet_recvd[i] = 0;
packet_forwarded[i] = 0;
packet_sent[i] = 0;
for(j=0;j<flows;j++) {
end_energy[i][j]    = 100.000000;
start_energy[i][j]=-1.00;}
```

```
end_time[i] = 10.000000000;
packet_end_time[i][0] = 10.000000000000;}
}

{
if (FILENAME == "nodes.txt"){
cnxs++;
sender_node[cnxs]=$0;
}
else{
state           =       $1;
time            =       $3;
flow            =       $39;

node_num        =       $5;
energy_level    =       $17;

pkt_size        =       $37;
energy_left_N   =       $7;
node_id         =       $9;
level           =       $19;
pkt_type        =       $35;
packet_id       =       $41;
no_of_forwards  =       $51;

if(state != "N") {


if ((state == "r" ) && (node_id == 0)) {
pkts_0_rcvd_size[packet_id] = pkt_size;
pkts_0_rcvd_time[packet_id] = time;}

if((pkt_type == "tcp") && (state == "s") && (level=="AGT")) {
packet_sent[flow]++;
packet_start_time[flow][packet_id] = time;
sender_node[flow] = node_id;
if(time < start_time[flow] ) start_time[flow] = time;
}
else if((pkt_type == "tcp") && (state == "r") && (level =="AGT")) {
packet_recvd[flow]++ ;
packet_end_time[flow][packet_id] = time;
end_time[flow] = time;
total_rcvd_byte[flow] = total_rcvd_byte[flow] + pkt_size;
if ( no_of_forwards > max_hop_count[flow] )
max_hop_count[flow] = no_of_forwards;
total_hop_count[flow] = total_hop_count[flow] + no_of_forwards;
}
else if((pkt_type == "tcp") && (state == "d")) { packet_drop[flow]++;}
else if((pkt_type == "tcp") && (state == "f")) { packet_forwarded[flow]++;}

if((level == "RTR") && (pkt_type == "AODV")){
aodv_rmsg = $61;
if((state == "r") || (state == "s")){
if (aodv_rmsg == "REQUEST"){ aodv_rreq[node_id]++;
}else aodv_rrep[node_id]++;
}#else if (state == "d") aodv_drop[node_id]++;
}


if (energy_level > start_energy[flow][node_id])
start_energy[flow][node_id] = energy_level;
if (energy_level < end_energy[flow][node_id])
end_energy[flow][node_id] = energy_level;

if ((state == "r") && (node_id == 0)) rec[packet_id] = time;

}
}
}
END {
```

```
#cnxs++;
for(j=0;j<=cnxs;j++) {
for(i=0;i<70;i++) {
if (start_energy[j][i] > end_energy[j][i])
total_energy_consumed[j] += (start_energy[j][i] - end_energy[j][i]);
}
}

nrg = total_energy_consumed[0]/cnxs;

for(i=1;i<=cnxs;i++) {
sum=0.000000000;
recvnum=0;
packet_duration=0.000000000;

for ( j in pkts_0_rcvd_size) {
if ((pkts_0_rcvd_time[j] >= start_time[i]) && (pkts_0_rcvd_time[j] <= end_time[i]))
tot_0_rcvd_pkts_size[i] += pkts_0_rcvd_size[j];
}

for ( j in rec) {
if ((rec[j] >= start_time[i]) && (rec[j] <= end_time[i])) all_packets[i]++;
}

if (packet_sent[i] != 0){
pkt_delivery_ratio[i] = (packet_recvd[i]/packet_sent[i])*100;
}

for ( j in packet_end_time[i] ) {
start = packet_start_time[i][j];
end = packet_end_time[i][j];
packet_duration = end - start;
if ( packet_duration > 0 )  { sum += packet_duration; recvnum++; }
}

if (recvnum!=0){delay[i]=sum/recvnum;}

file_name = "weka_file_" cnxs ".arff"
duration = end_time[i]-start_time[i];

printf("%.8f,", duration) >> (file_name);
printf("%d,", all_packets[i]) > (file_name);
printf("%d,", aodv_rreq[sender_node[i]]) > (file_name);
printf("%d,", aodv_rrep[sender_node[i]]) > (file_name);
printf("%d,", packet_sent[i]) > (file_name);
printf("%d,", packet_recvd[i]) > (file_name);
printf("%d,", packet_drop[i]) > (file_name);
printf("%d,", packet_forwarded[i]) > (file_name);
printf("%.6f,",total_energy_consumed[i] + (nrg * duration)) > (file_name);
printf("%.2f,",pkt_delivery_ratio[i]) > (file_name);
printf("%.9f,",delay[i]) > (file_name);
printf("%d,", max_hop_count[i]) > (file_name);
printf("%d,", avg_hop_count[i]) > (file_name);
printf("%.4f,", (((tot_0_rcvd_pkts_size[i]*8)/duration)/1024)) > (file_name);

if (FILENAME == "Normal.tr"){printf("Normal\n") > (file_name);}
else if (FILENAME == "Blackhole.tr"){printf("Blackhole\n")  > (file_name);}
else if (FILENAME == "DoS.tr"){printf("DoS\n")  > (file_name);}
else if (FILENAME == "Flood.tr"){printf("Flood\n")  > (file_name);}


}
}
```

References

[1] Security tip (st04-015) understanding denial-of-service attacks. 17

[2] Buyers guide to intrusion detection and prevention tools, Aug 2016. 19

[3] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. Wireless sensor networks: a survey. *Computer networks 38*, 4 (2002), 393–422. 5

[4] Ali, M. Z. A robust user authentication scheme for wireless sensor network. 10

[5] Altman, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician 46*, 3 (1992), 175–185. 31

[6] Anchugam, C., and Thangadurai, K. Security in wireless sensor networks (wsns) and their applications. In *Information Fusion for Cyber-Security Analytics*. Springer, 2017, pp. 195–228. 8

[7] Animesh P, J.-M. P. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Science Direct, Computer Networks 51* (2007), 3448–3470. 21

[8] Anu B, Devi R, L. K. R. M. S. N. Intrusion detection system using eaack algorithm. *International Journal of Advance Research in Science and Engineering 5* (April 2016), 566,573. 17

[9] Bakht, H., et al. Survey of routing protocols for mobile ad-hoc network. *International Journal of Information and Communication Technology Research 1*, 6 (2011). 13

[10] Barros, M. T., Gomes, R. C., de Alencar&, M. S., and Costa, A. F. Feature filtering techniques applied in ip traffic classification. 38

[11] Becker, B. G. Visualizing decision table classifiers. In *Information Visualization, 1998. Proceedings. IEEE Symposium on* (1998), IEEE, pp. 102–105. 31

[12] Beigh, B. M., Bashir, U., and Chahcoo, M. Intrusion detection and prevention system: Issues and challenges. *International Journal of Computer Applications 76*, 17 (2013). 18

[13] Brutch, P., and Ko, C. Challenges in intrusion detection for wireless ad-hoc networks. In *Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on* (2003), IEEE, pp. 368–373. 2

[14] Butun, I., Morgera, S. D., and Sankar, R. A survey of intrusion detection systems in wireless sensor networks. *IEEE Communications Surveys & Tutorials 16*, 1 (2014), 266–282. 2, 16, 22

[15] Chen, X., Makki, K., Yen, K., and Pissinou, N. Sensor network security: a survey. *IEEE Communications Surveys & Tutorials 11*, 2 (2009). 1

[16] Chin, K.-W., Judge, J., Williams, A., and Kermode, R. Implementation experience with manet routing protocols. *ACM SIGCOMM Computer Communication Review 32*, 5 (2002), 49–59. 13

[17] Darra, E., Skouloudi, C., and Katsikas, S. K. A simulation platform for evaluating dos attacks in wireless sensor networks. In *Proceedings of the 19th Panhellenic Conference on Informatics* (2015), ACM, pp. 144–149. 24

[18] De Boer, P., and Pels, M. Host-based intrusion detection systems. *Amsterdam University* (2005). 18

[19] Denning, D. E. An intrusion-detection model. *IEEE Transactions on software engineering*, 2 (1987), 222–232. 16

[20] Dickerson, J. E., and Dickerson, J. A. Fuzzy network profiling for intrusion detection. In *Fuzzy Information Processing Society, 2000. NAFIPS. 19th International Conference of the North American* (2000), IEEE, pp. 301–306. 22

[21] Dini, G., and Tiloca, M. On simulative analysis of attack impact in wireless sensor networks. In *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on* (2013), IEEE, pp. 1–8. 25

[22] El Kateeb, A., Ramesh, A., and Azzawi, L. Wireless sensor nodes processor architecture and design. In *Advanced Information Networking and Applications-Workshops, 2008. AINAW 2008. 22nd International Conference on* (2008), IEEE, pp. 892–897. 6

[23] Ganz, F. M., and Burkhardt, D. P. Wireless burstable communications repeater, June 24 2003. US Patent 6,584,080. 8

[24] Ghorbani, A. A., Lu, W., and Tavallaee, M. Detection approaches. *Network Intrusion Detection and Prevention* (2010), 27–53. 23

[25] Gite, P., and Thakur, S. Different security issues over manet. *International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR) 1*, 3 (2013), 233–238. 17

[26] Grover, J., and Sharma, S. Security issues in wireless sensor network—a review. In *Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO), 2016 5th International Conference on* (2016), IEEE, pp. 397–404. 1

[27] Han, C.-C., Kumar, R., Shea, R., Kohler, E., and Srivastava, M. A dynamic operating system for sensor nodes. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services* (2005), ACM, pp. 163–176. 8

[28] Ho, T. K. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on* (1995), vol. 1, IEEE, pp. 278–282. 31

[29] Hu, Y.-C., Perrig, A., and Johnson, D. B. Packet leashes: a defense against wormhole attacks in wireless networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies* (2003), vol. 3, IEEE, pp. 1976–1986. 2

[30] Jiménez, V. P. G., and Armada, A. G. Field measurements and guidelines for the application of wireless sensor networks to the environment and security. *Sensors 9*, 12 (2009), 10309–10325. 14

[31] Jyothsna, V., Prasad, V. R., and Prasad, K. M. A review of anomaly based intrusion detection systems. *International Journal of Computer Applications 28*, 7 (2011), 26–35. 21

[32] K. S. Low, H. A. N., and Guo, H. Optimization of sensor node locations in a wireless sensor network. *International Conference on Natural Computation 5* (2008), 286,290. 7

[33] Kabir, M. H., Islam, S., Hossain, M. J., and Hossain, S. Detail comparison of network simulators. *International Journal of Scientific & Engineering Research 5* (2014), 203–218. 33

[34] Kachirski, O., and Guha, R. Effective intrusion detection using multiple sensors in wireless ad hoc networks. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on* (2003), IEEE, pp. 8–pp. 18

[35] Karlof, C., Sastry, N., and Wagner, D. Tinysec: a link layer security architecture for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems* (2004), ACM, pp. 162–175. 1

[36] Karlof, C., and Wagner, D. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad hoc networks 1*, 2 (2003), 293–315. 1, 2

[37] Kaushal, K., and Sahni, V. Dos attacks on different layers of wsn: A review. *International Journal of Computer Applications 130*, 17 (2015). 17

[38] Knight, C., Davidson, J., and Behrens, S. Energy

options for wireless sensor nodes. *Sensors 8*, 12 (2008), 8037–8066. 7

[39] Labib, K. Computer security and intrusion detection. *Crossroads 11*, 1 (2004), 2–2. 16

[40] Lee, G.-J., Kong, J.-U., Lee, M.-S., and Byeon, O.-H. A cluster-based energy-efficient routing protocol without location information for sensor networks. *Journal of Information Processing Systems 1*, 1 (2005), 49–54. 11

[41] Lin, Y., Zhang, Y., and Ou, Y.-j. The design and implementation of host-based intrusion detection system. In *Intelligent Information Technology and Security Informatics (IITSI), 2010 Third International Symposium on* (2010), IEEE, pp. 595–598. 18

[42] Logambal, R., and Chitra, K. A study on routing protocols for mobile adhoc networks". *International Journal of Innovative Research in Computer and Communication Engineering 2*, 3 (2004). 12

[43] Markou, M., and Singh, S. Novelty detection: a review—part 2:: neural network based approaches. *Signal processing 83*, 12 (2003), 2499–2521. 20

[44] Messai, M.-L. Classification of attacks in wireless sensor networks. *arXiv preprint arXiv:1406.4516* (2014). 17

[45] Methley, S. *Essentials of wireless mesh networking*. Cambridge University Press, 2009. 12

[46] Minaie, A., Sanati-Mehrizy, A., Sanati-Mehrizy, P., and Sanati-Mehrizy, R. Application of wireless sensor networks in health care system. *age 23* (2013), 1. 1

[47] Mukherjee, B., Heberlein, L. T., and Levitt, K. N. Network intrusion detection. *IEEE network 8*, 3 (1994), 26–41. 18

[48] Mv °ller, K.-R., Mika, S., Rv § tsch, G., Tsuda, K., and Schva lkopf, B. An introduction to kernel-based learning algorithms. In *Handbook of Neural Network Signal Processing*. CRC Press, 2001. 31

[49] Narayana, M. S., Prasad, B., Srividhya, A., and Reddy, K. P. R. Data mining machine learning techniques–a study on abnormal anomaly detection system. *International Journal of Computer Science and Telecommunications 2*, 6 (2011). 22

[50] Oliveira, L. M., and Rodrigues, J. J. Wireless sensor networks: A survey on environmental monitoring. *JCM 6*, 2 (2011), 143–151. 10

[51] Perrig, A., Szewczyk, R., Tygar, J. D., Wen, V., and Culler, D. E. Spins: Security protocols for sensor networks. *Wireless networks 8*, 5 (2002), 521–534. 1

[52] Purniemaa, P., Manikandan, K., and Durai, M. S. A review on security issues in multipath routing protocol in manet. *International Journal of Advanced Research in Computer Science 2*, 3 (2011). 14

[53] Purohit, R., and Mathur, P. Role of wireless sensor networks in communication with artificial intelligence system. 5

[54] Qayyum, A., Islam, M., and Jamil, M. Taxonomy of statistical based anomaly detection techniques for intrusion detection. In *Emerging Technologies, 2005. Proceedings of the IEEE Symposium on* (2005), IEEE, pp. 270–276. 21

[55] Stojmenovic, I. *Handbook of sensor networks: algorithms and architectures*, vol. 49. John Wiley & Sons, 2005. 28

[56] Stutz, M. Get started with gawk: Awk language fundamentals, September 2006. 34

[57] Sun, B., Osborne, L., Xiao, Y., and Guizani, S. Intrusion detection techniques in mobile ad hoc and wireless sensor networks. *IEEE Wireless Communications 14*, 5 (2007). 2, 24

[58] Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., and Lin, W.-Y. Intrusion detection by machine learning: A review. *Expert Systems with Applications 36*, 10 (2009), 11994–12000. 2, 22

[59] Đurišić, M. P., Tafa, Z., Dimić, G., and Milutinović, V. A survey of military applications of wireless sensor networks. In *Embedded Computing (MECO), 2012 Mediterranean Conference on* (2012), IEEE, pp. 196–199. 1

[60] Vail, D., and Veloso, M. Learning from accelerometer data on a legged robot. In *Proceedings of the 5th IFAC/EURON symposium on intelligent autonomous vehicles* (2004). 31

[61] Whitaker, A., and Newman, D. P. *Penetration testing and network defense*. Cisco Press, 2005. 17

[62] Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016. 35

[63] Wood, A. D., and Stankovic, J. A. Denial of service in sensor networks. *computer 35*, 10 (2002), 54–62. 1

[64] Yu, Z., and Tsai, J. J. A framework of machine learning based intrusion detection for wireless sensor networks. In

Sensor Networks, Ubiquitous and Trustworthy Computing, 2008. SUTC'08. IEEE International Conference on (2008), IEEE, pp. 272–279. 2