



SAPIENZA  
UNIVERSITÀ DI ROMA

# Machine Learning Techniques for Intrusion Detection in Internet of Things Networks

Facoltà di Ingegneria dell' Informazione, Informatica e Statistica  
Corso di Laurea Magistrale in Engineering in Computer Science

Candidate

Federico Bacci

ID Number 1609616

Thesis Advisor

Prof. Ioannis Chatzigiannakis

Academic Year 2018 / 2019



Machine Learning Techniques for Intrusion Detection in Internet of Things Networks

Sapienza – University of Rome © 2019 Federico Bacci.

Version: May 18, 2019 Author's email: [fedeb703@gmail.com](mailto:fedeb703@gmail.com)

# Contents

<b>Contents</b>	<b>3</b>
<b>Abstract</b>	<b>5</b>
<b>Introduction</b>	<b>6</b>
Document Structure	6
Chapter 1: Internet of Things and Security Challenges	6
Chapter 2: Use Cases and Data Creation	6
Chapter 3: Data analysis	7
Chapter 4: Conclusions and Future Work	7
<b>Chapter 1: Internet of Things and Security Challenges</b>	<b>8</b>
Internet of Things	8
Consumer Applications	11
Commercial Applications	12
Infrastructure Applications	13
Industry Applications	13
Industry 4.0	14
Impact of Industry 4.0	15
Challenges of Industry 4.0	16
Security Issues	16
Threats Types	17
Wireless Threats	17
Routing Threats	17
Denial of Service (DoS)	18
Security Solutions	18
Machine Learning Techniques	19
Supervised Learning	20
Unsupervised Learning	22
Main Concept of this Thesis	24
<b>Chapter 2: Use Cases and Data Creation</b>	<b>26</b>
Real Use Case: IOT-LAB	26
Simulated Use Case: Cooja Simulator	27
Contiki Features	28
Data Types	36
<b>Chapter 3: Data analysis</b>	<b>39</b>
<b>Tools</b>	<b>39</b>
Libraries	39

System	39
Github Repository	40
Phase 1: Data Analysis per Node	40
Phase 2: Generating per Node Statistics	43
Phase 3: Machine Learning Techniques	51
Phase 4: Accuracy of the method	55
<b>Chapter 4: Conclusions and Future Work</b>	<b>57</b>
Conclusions	58
Future work	58
<b>References</b>	<b>60</b>

# Abstract

More and more embedded processor are activated everyday, most of them to industrial equipment to create the fourth industrial revolution.

These embedded processor sense and process data to create insights enabling smart factories that can operate without physical intervention opening the way to new possibilities and new challenges, one of all is the security of the data and of the networks of these industries.

Despite the compelling features of Industrial Internet of Things, the security of such network is impeding their rapid deployment.

In this thesis we try to use machine learning to analyze the Intrusion detection in such networks (IPv6 based) using data from both simulated and real world deployment of Internet of Things Networks.

We propose a data-driven anomaly detection that operates at transport layer of 6LoWPAN deployments and exploring the possibilities of different tools.

# Introduction

## Document Structure

- Chapter 1: Internet of Things and Security challenges
- Chapter 2: Use case and definition for creation of data
- Chapter 3: Data Analysis
- Chapter 4: Conclusion and future work

## Chapter 1: Internet of Things and Security Challenges

To create and work on this thesis was necessary an introduction on the topic and on all of component used.

Here it is presented the research done to approach and better understand the methods and the solution proposed.

- Internet of Things
- Industry 4.0
- Security Issues & vulnerabilities
- Intrusion types
- Machine Learning Techniques

Going through all the problem faced while talking about Internet of Things and Security of the network.

## Chapter 2: Use Cases and Data Creation

To find a solution and validate our first hypothesis a Use Case is needed, here is described the different approaches used to create both use cases and data to better understand how Internet of things works on the Transport layer.

## **Chapter 3: Data analysis**

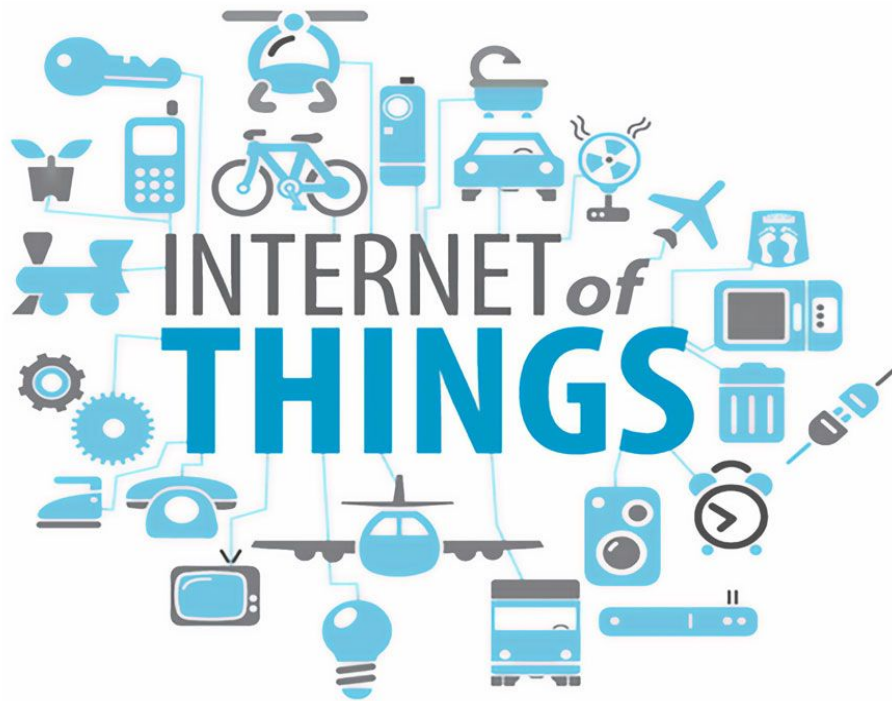
After the creation of the data is needed to analyze them in order to extrapolate important information on the different behaviour of the nodes and the network in both Normal Case and Malicious Case using different Machine Learning Techniques.

## **Chapter 4: Conclusions and Future Work**

In this chapter will be elaborated the conclusions on the proposed system, suggesting the implementation and continue the work of the thesis to create a real and usable product.



# Chapter 1: Internet of Things and Security Challenges



# Internet of Things

The Internet of Things (IoT) is an extension of internet connectivity into physical devices and everyday objects. These devices can be embedded into any forms of hardware and can communicate and interact over internet other than being monitored and controlled. [16][17][18][19][20]

In 1994, Reza Raji described the concept of IoT as “moving small packets of data to a large set of nodes so as to integrate and automate everything from home appliance to entire factories”[23].

The term IoT was born in 1999 [24] with (RFID) as essential to Internet of things to allow computers to manage all individual things.

In 2010 there were already 1.84 connected devices per person and by 2020 there will be 50 billion devices.

But which are the major components of Internet of Things?

# Major Components of IoT



## Smart devices and sensors

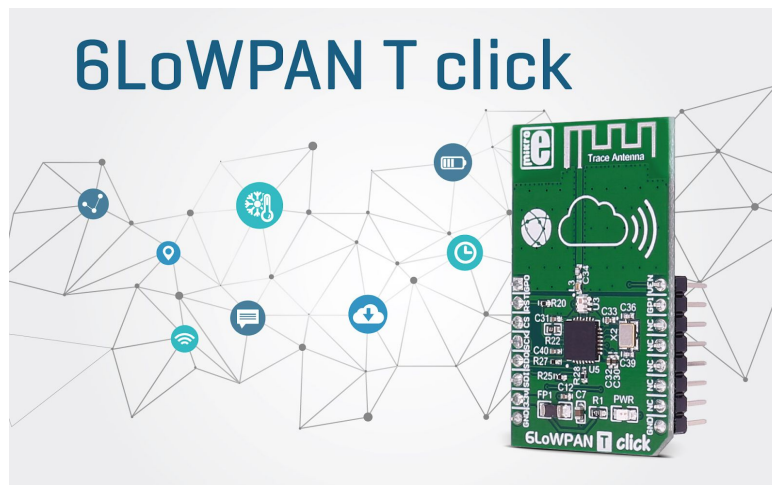
These are sensors capable of collect data from the environment and transmit the informations to the next layer.

These sensors are connected through Low power and lossy network like:

- WiFi
- ZigBee
- bluetooth
- Z-wave

Developments in the low power, low cost wireless transmitting devices are promising in the area of IoT due to its long battery life and efficiency.[21]

The latest protocol like **6LowPAN-IPv6** [30] (Internet Protocol v6 over Low Power Wireless Personal Area Networks) has defined encapsulation to enable the packets to be sent and received over IEEE 802.15.4



The specific of 6LoWPAN proposed by the working group are an adaptation of IPv6 packets over IEEE 802.15.4 as data-link and physical layer protocol. 6LoWPAN uses only UDP as TCP is considered to be too resource-demanding.

The **IPv6 Routing Protocol for Low-Power and Lossy Networks** (RPL) [32] is a standardized routing protocol primarily used in a 6LoWPAN network. RPL creates a **destination-oriented directed acyclic graph (DODAG)** between the nodes in a 6LoWPAN.

RPL supports traffic towards a DODAG root and bidirectional traffic between 6LoWPAN devices and between devices and the DODAG root.

There may exist multiple global RPL instances for a single 6LoWPAN network and a local RPL DODAG can be created among a set of nodes inside a global DODAG:

Each node in a DODAG has a rank that indicates the position of a node relative to other nodes and with respect to the DODAG root. Ranks strictly decrease in the up direction towards the DODAG root and strictly increase from the DODAG root towards nodes.

The RPL protocol provides new ICMPv6 control messages [39] to exchange routing graph information.

## Gateway

IoT **Gateway manages the bidirectional data** traffic between different networks and protocols. Another function of gateway is to translate different network protocols and make sure interoperability of the connected devices and sensors.

Gateways can be configured to perform pre-processing of the collected data from thousands of sensors locally before transmitting it to the next stage. In some scenarios, it would be necessary due to compatibility of TCP/IP protocol.

IoT gateway offers certain level of security for the network and transmitted data with higher order encryption techniques. It acts as a middle layer between devices and cloud to protect the system from malicious attacks and unauthorized access.

## Cloud

Internet of things creates massive data from devices, applications and users which has to be managed in an efficient way. IoT cloud offers tools to collect, process, manage and store huge amount of data in real time. Industries and services can easily access these data remotely and make critical decisions when necessary.

Basically, **IoT cloud is a sophisticated high performance network of servers optimized to perform high speed data processing of billions of devices**, traffic management and deliver accurate analytics. Distributed database management systems are one of the most important components of IoT cloud.

Cloud system integrates billions of devices, sensors, gateways, protocols, data storage and provides predictive analytics. Companies use these analytics data for improvement of products and services, preventive measures for certain steps and build their new business model accurately.

## **Analytics**

One of the major Advantages of an efficient IoT System is real time **smart analytics** which helps engineers to find out irregularities in the collected data and act fast to prevent and undesired scenario.

Service providers can prepare for further steps if the information is collected accurately at the right time.

Information is very significant in any business model and predictive analysis ensures success in concerned area of business line.

## **User Interface**

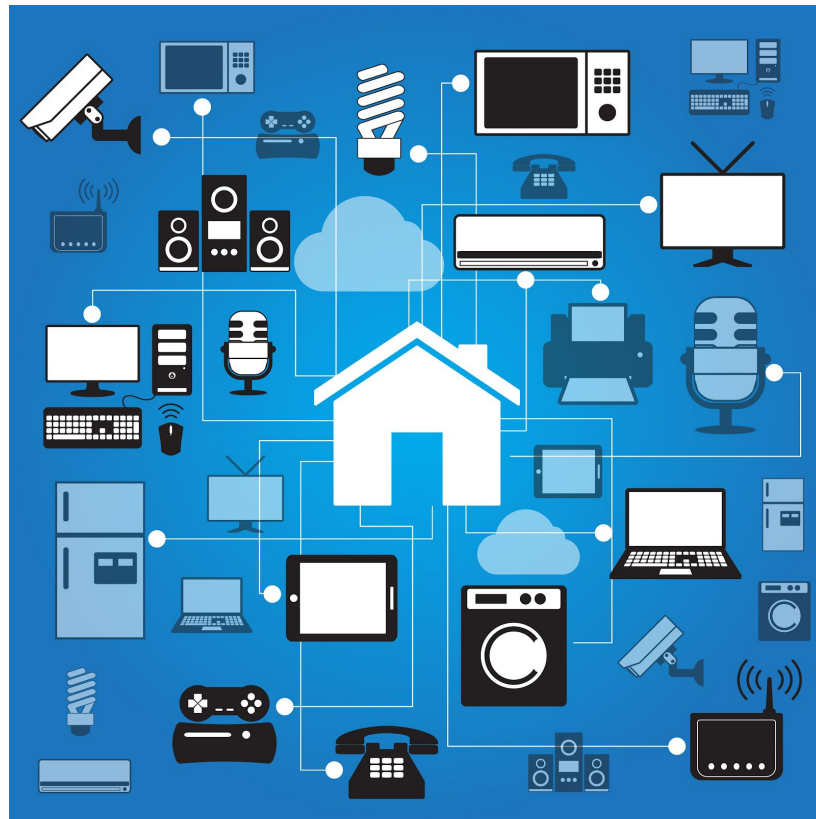
User Interfaces are the visible, tangible part of the IoT system which can be accessed by users. Modern technology offers much interactive design to ease complex tasks into simple touch panels control. These panels have replaced hard switches in appliances and the trend is increasing.

Having more and more set of application they can be divided them into 4 groups

- Consumer
- Commercial
- Infrastructure Spaces
- Industrial

## **Consumer Applications**

In this group there are all the IoT devices for consumer use such as home automation, wearable technology, connected health and so on.[40][41]



A lot of focus is in the last 5 years and in the last one to smart home hub to control smart home devices such as



- Heating, ventilation and air conditioning
- Lighting control system
- Smart grid and control meter
- Home robots
- Leak, smoke and CO detector
- Indoor positioning system
- Air control
- Smart Kitchen

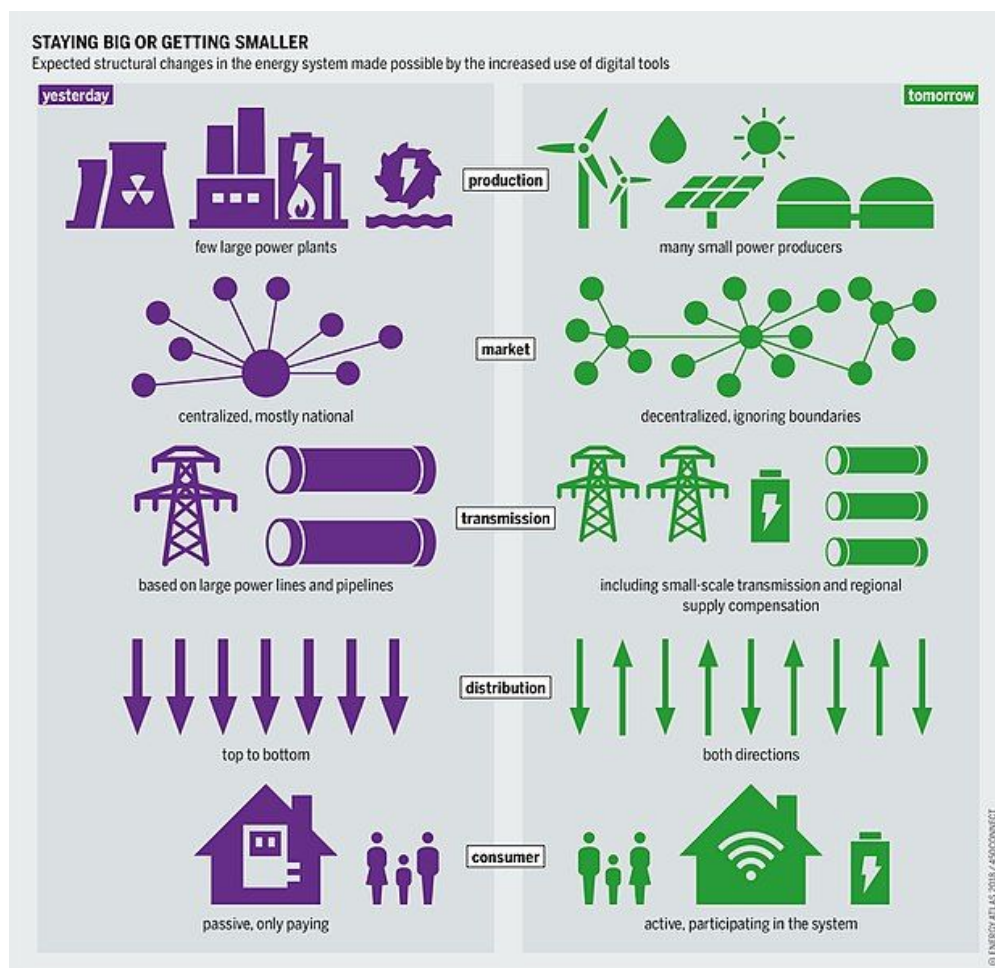
These hubs are used to control all IoT devices in the home and usually are voice or touch controlled such as Google Home or Amazon Echo .

## Commercial Applications

In this group there are IoT devices intended for Medical and Healthcare, Transportation and Building Purposes such as Smart Heart monitors for hospitals [15] or Smart Traffic control and Smart Parking devices [3] and Smart buildings[2] for example schools ones[13]

This is still a new field and is expected to grow in the next years .

## Infrastructure Applications



Monitoring and controlling operation of sustainable urban and rural infrastructures like bridges, railway tracks and wind-farms is a key application for IoT [6][7].

The use of IoT in this case benefits with cost saving, time reduction and increase productivity with Real-Time Data Analytics.

Using IoT devices for monitoring and operating infrastructure will improve also incident management and the quality of service even in areas such as waste management.

With IoT devices Smart grids [26] can be created where both consumer and producer benefits from the data gathered by IoT devices implemented for example in the case of the smart grid created by Tesla in Australia.

## Industry Applications

**Industrial Internet of Things (IIOT)** refers to interconnected sensors, instruments and other devices networked together with computers' industrial applications. [42]



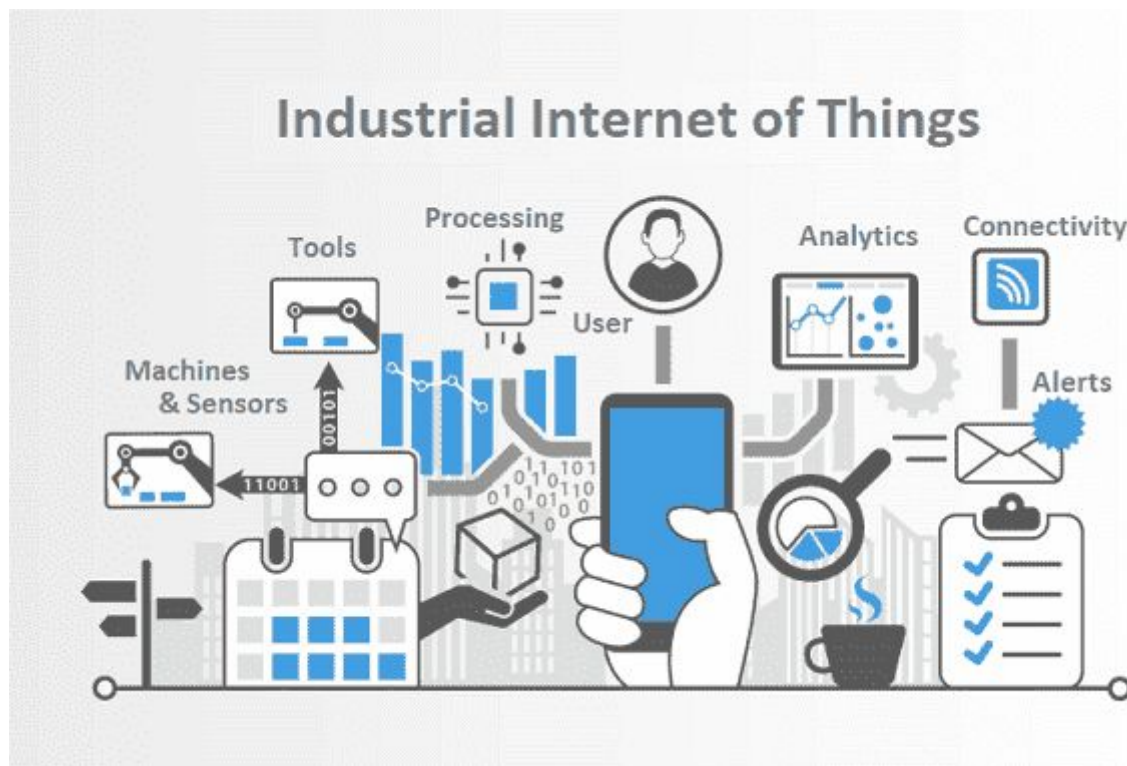
This kind of application is very important because highly integrated smart cyber physical space opens door to create a whole new business and market opportunities for manufacturing.

The key goal of IIOT is to extend traditional Service-Oriented Architectures (SOA) that are predominantly used within industrial computing and networking infrastructures, into the embedded world, where digital representations of real world services with traditional services and data.

IoT intelligent systems enables:

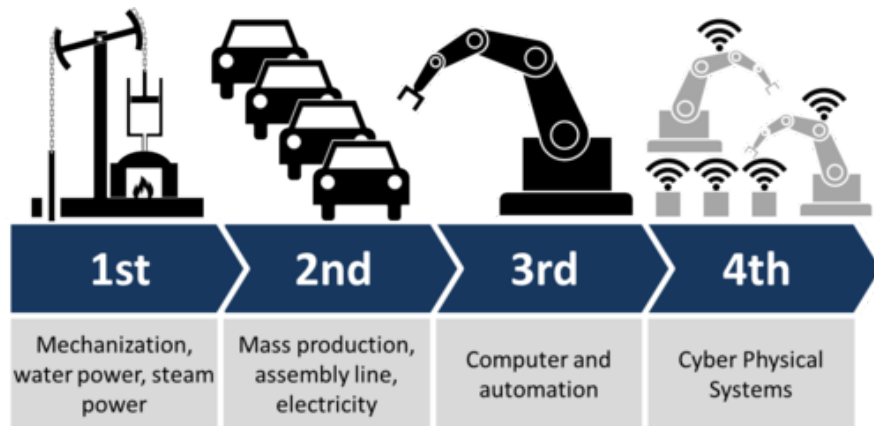
- rapid manufacturing of new products
- dynamic response to product demands
- real time optimization of manufacturing
- Predictive maintenance
- Statistical evaluation
- Maximize the reliability

The term Industrial Internet of Things (IIOT) is often encountered in manufacturing industries.



## Industry 4.0

The Industrial Internet of things could generate so much business value that it will lead to the Fourth Industrial Revolution also called Industry 4.0. [43]



The term Industry 4.0 (I4) originates from a project in the high-tech strategy of the German Government, revived then in 2011, during 2013 at the Hannover Fair the final report of the Working Group Industry 4.0 was presented.

There are four design principles in Industry 4.0:

- **Interconnection:** The ability of machines, devices, sensors and people to connect and communicate via IoT
- **Information Transparency:** the technology should provide transparency, Inter connectivity allow operator to collect immense amount of data and information from all points, aiding the functionality to identify areas that can benefit from improvement
- **Technical assistance:** The ability of assistance system to aggregating and visualising information comprehensively for making decision. Also the ability of cyber physical system to support humans conducting task unsafe, exhausting and unpleasant.
- **Decentralized decision:** ability to make decision autonomously directly by the systems.

## Impact of Industry 4.0

The fourth Industrial revolution has a huge impact on:

- Services and business models
- Reliability and continuous productivity
- IT Security
- Machine Safety
- Manufacturing Sales
- Product lifecycle
- Industry value chain
- Workers education and skills
- Socio-economic factors



## Challenges of Industry 4.0

Along with impacts there are challenges to be faced the challenges of this revolution

- Integrity of production
- Need for skills
- IT Security issues
- Data Security
- Protection of industrial secrets

## Security Issues

As enlightened from the previous section one of the **main challenges** both in Internet of Things and in the Industry 4.0 is the Security and Vulnerabilities.

In fact as more IoT infrastructures are deployed and smart city services become integral part of our lives, these issues need to be addressed.[28][7]

Security issues can be summarized in different categories:

- Security
- Privacy
- Trust

The high number of interconnected devices arises also scalability issues, so a flexible infrastructure is needed to be able to deal with security threat in such a dynamic environment.[1]

The vision of the IoT has led to a competitive market for stakeholders that, in the absence of common standards, market proprietary and application-specific solutions, including a variety of hardware platform, operating systems, communication protocols and data management schemes.

Now the deployment of IIOT are closed source and privately runned, these application-specific deployments have limited scope in data while information and knowledge sharing is achieved through custom internet gateways. These solution are non-scalable, with low cost efficiency and non adaptive.

Different bodies are trying to standardise but no standard till now has managed to attract the vast majority of the stakeholders.

IIOT systems are characterized by energy constraints, irregular configurations, time-varying topology, large scale and changing applications. thus the solutions need to be differently implemented respect the ad-hoc networking.

In sensors networks:

- The number of interacting devices is extremely large and dense
- The resources are very limited
- There is no fixed infrastructure
- Network topology is unknown before deployment
- High risk of physical attacks

While in traditional wired and wireless networks these are standard security requirements in low cost wireless sensors are more vulnerable.

For that reason an adversary can easily capture the devices, read the content of the memory learning the cryptographic secret.

In addition the high node-to-human ratio make impossible to consider the presence of an online server that maintains individual nodes constantly, making impossible techniques of pre distribution of keys.

Adding to that nodes are battery operated, so security systems must reduce the energy consumption. IOT devices have also limited computing and storage capability, so cryptographic algorithms and protocols that require intensive computation, communication or storage are not applicable.

An IIOT node is defined a normal node or node with normal behaviour if it operates following the systems specifications. Otherwise it is catalogued it as malicious node or adversary.

## Threats Types

### Wireless Threats

The most basic threats are due to communication that takes places over a wireless channel. Wireless communication suffers from a number of vulnerabilities:[44]

- **Eavesdropping** The most easy way is to overhear the information that the node transmits or receives and then analyze the data and extract sensitive information without interacting with the network. This is part of Passive attacks.
- **Data alteration** An adversary can cause collisions of wireless transmissions and then try to modify the message exchanged between wireless parties. This is part of Active attacks.
- **Identity Theft** The attacker can impersonate a legitimate user and when the original user is inactive, transmit messages without being noticed.

### Routing Threats

As many routing protocol for wireless sensor are simple they can be an easy target for attacks.

Attacks of this type can be classified in:

- **Spoofed, altered, or replayed routing information** while sending the data the information in transit could be altered, spoofed, replayed or destroyed. Since the nodes have a short range transmission, an attacker with high processing power and

large communication range could attack several sensors simultaneously and modify the transmitted informations.

- **Selective forwarding** The malicious node may refuse to forward every message it gets, acting as black hole or just forwarding part of them acting as a Gray Hole
- **Sinkhole Attacks** In this case the goal of the attacker is to attract all of the traffic especially in the case of a flooding based protocol. Then the malicious node can listen, to request for routes and then reply to the requesting node with messages containing a bogus route with the shortest path to the requested destination
- **Sybil Attacks** In Sybil attack the compromised node presents itself as multiple nodes, as a effect the usage and the efficiency of the distributed algorithms are degraded. Sybil attack can be made against distributed storage, routing, fair resource allocation and misbehavior detection
- **Wormholes** The malicious node tunnels messages from one part of the network over a link that doesn't exist normally to other part of the network. The simplest form of the wormhole attack is to convince two nodes that are neighbors. This can be used in combination with selective forwarding and eavesdropping
- **HELLO flood attacks** this attack is based on the use by many protocol of broadcast Hello messages to announce themselves to the network. An attacker with greater transmission may send hello messages to a large number of nodes, convincing them that the attacker is near. This leaves the network in a confusion state
- **Acknowledgement** Some network of IIOT require link layer acknowledgements. A compromised node may exploit this by spoofing these acknowledgements, convincing the sender that a weak link is strong or a dead sensor is alive

## Denial of Service (DoS)

This class of attack does not concern the information on the network, rather the goal of the attacker is to exhaust the resource of the network until it does not function properly.

In physical layer these could be jamming and tampering. Jamming is done by interfering with the radio frequencies used by the node. Tampering refers to damaging and altering the nodes.

An attacker can damage and replace a node stealing or replacing information and cryptographic keys.

## Security Solutions

Some solution proposed till now focus on the validation of the integrity of message exchanges in order to securely route information across the network, other try to solve the problem proposing a secure group of communication primitives on top of the networking layer.[45]

# Machine Learning Techniques

*"The real value that the Internet of Things creates is at the intersection of gathering data and leveraging it. All the information gathered by all the sensors in the world isn't worth very much if there isn't an infrastructure in place to analyze it in real time."*

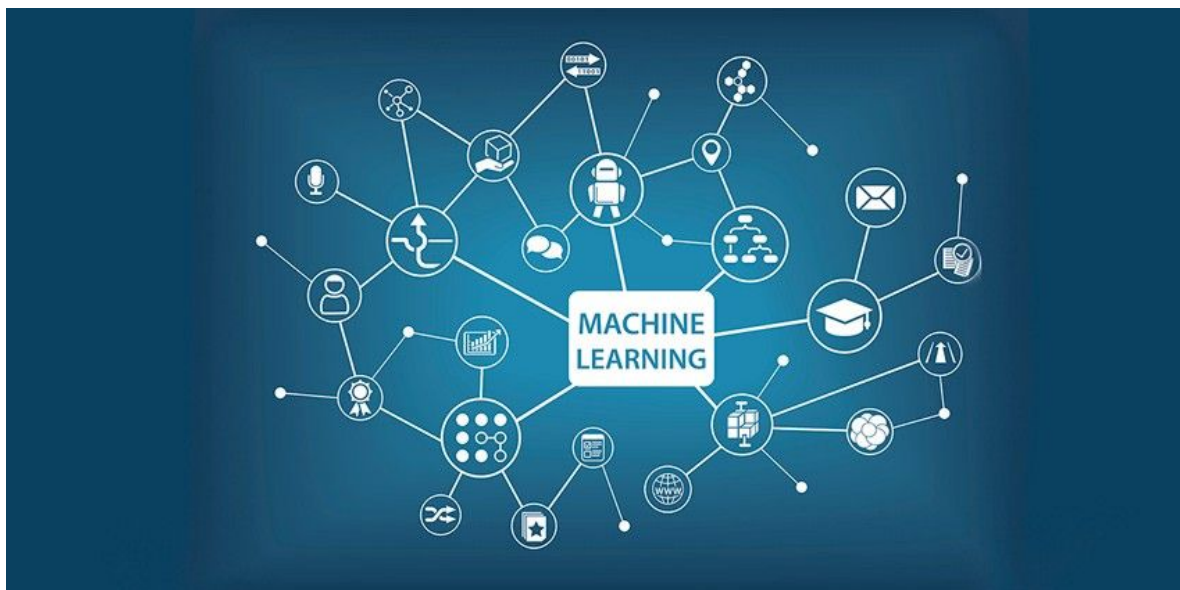
Wired Magazine, April 18, 2016

Machine Learning and Industrial Internet of things are linked by the fact that the increasing number of devices produces a multitude of information, it is needed algorithm that can learn and find useful information in these data.

**Machine Learning (ML)** is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task, relying on patterns and inference instead.[46]

Due to the fact that the network itself produces a lot of data on the transport layer, one use of the data produced that proposed is to help network owners and participants to find out if the network has been attacked or tampered.

In this thesis it is explored the use of Machine Learning applied to transport layer data to find out security threats, in particular intrusion detection.



The machine Learning techniques used in this thesis of two types:

- Supervised Learning
- Unsupervised Learning

## Supervised Learning

These algorithm make use of a training set of input and output data. The algorithm learns the relationship between the input and the output in the training set and then uses the relationship to predict the output of new data.

Classification learning can predict the membership of a certain class.

To solve a problem with supervised learning one has to follow these steps:

1. Determine the type of the training example
2. Gather a training set, this training set needs to be representative of the real world use case
3. Determine the input feature representation of the training set. Typically the object is transformed into a feature vector containing all the features that are used
4. Determine the algorithm used
5. Run the algorithm to determine control parameter and adjusting them to optimizing the performance on a subset of the training set
6. Evaluate the accuracy of the learned function, testing it on a separate set from the training set

During this work different supervised learning algorithms are explored

- Random Forest
- KNN
- SVM

### Random Forest

This algorithm operates by constructing a multitude of decision trees at training time and outputting the class or mean of prediction of the individual trees.

Random forest can be used also to rank importance of variable in a regression or classification.

This algorithm is useful when we want quick results without worrying too much about the dataset.

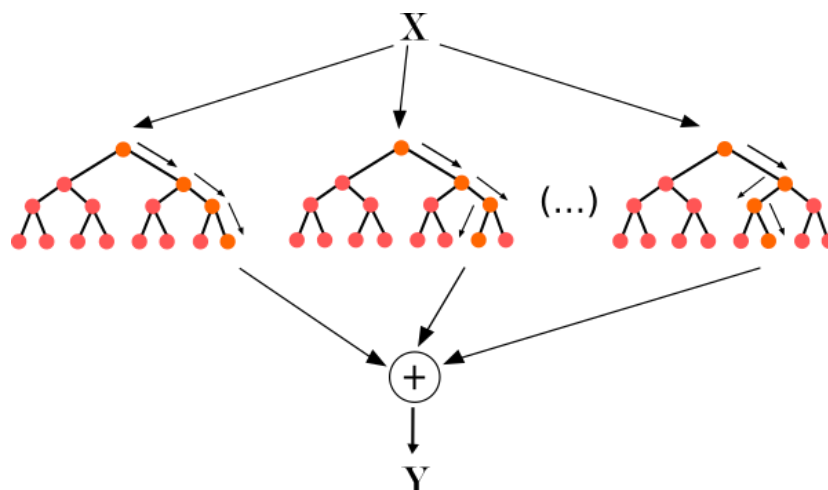


Fig 1 Example of Random Forest

### **K-Nearest Neighbors (KNN)**

is a nonparametric method used for classification and regression, the output could be a class membership or a property value for the object.

KNN is one of the most easy algorithms for machine learning.[47]

The training examples in this case are vector in a multidimensional feature space each with a class label,  $k$ , defined by the user, is classified by assigning the label which is most frequent among the  $k$  training examples nearest to that query point.

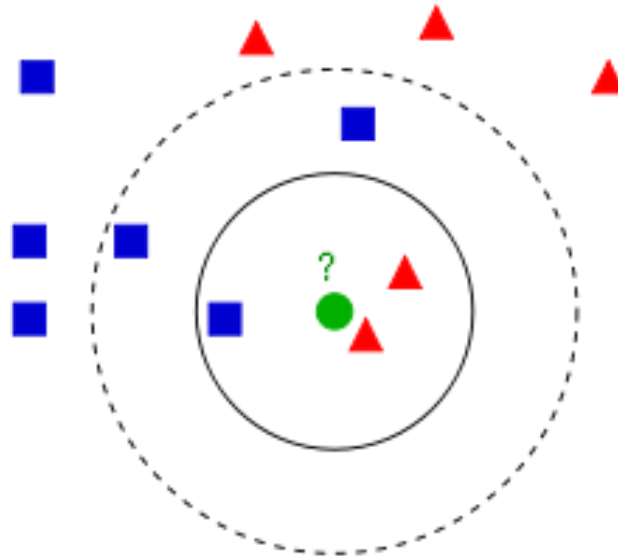


Fig 2 Example of KNN

### **Support Vector Machines (SVM)**

Given a set of training examples SVM training algorithms build a model that assign new example to one category or the others making it non probabilistic binary linear classifier.

Classifying data is a common task in machine learning, and in the case of SVM the point is viewed as  $p$  dimensional vector that can be separated with a  $(p-1)$  hyperplane.

One of the issue of this algorithm is that it requires full labelling of the data and it is directly applicable for two-class tasks, requiring for more complex task a multi class SVM algorithm.

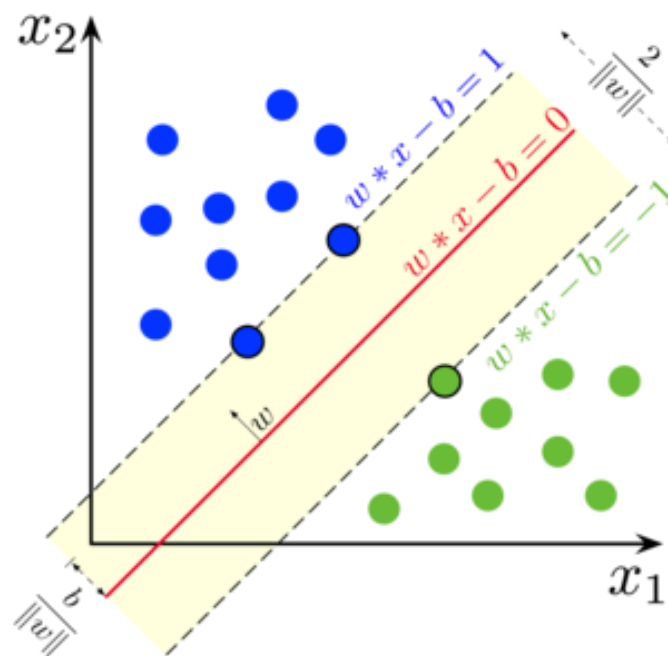


Fig 3 Example of SVM

## Unsupervised Learning

The unsupervised learning algorithm aims to make observations in data where there is no known outcome or results, deducing underlying pattern and structure in the data. Association learning is one of the most common forms, the algorithm searches for association between input data.

**K-Means** is a method of vector quantization very popular for cluster analysis.

K-Means aims to partition  $n$  observations into  $k$  clusters in which each observation belongs to the cluster with the nearest mean as prototype for the cluster.



Fig 4 Example of K means Algorithm

There are 4 steps to apply K Means [48][38]

1. **Initialization** The k number of data point is chosen randomly in the dataset (4 green points) as initial centroids, these will be the center of the cluster
2. **Cluster Assignment** All the data point that are the closest to a centroid will create a cluster, if the algorithm is using Euclidean distance, a straight line divides the two clusters, perpendicular to the line between two centroids.
3. **Move to centroid** after having the new clusters, a centroid is a new value that is the mean of all examples of the cluster
4. **Repeat** The steps 2 and 3 are repeated until the centroid don't change anymore



```

randomly chose k examples as initial centroids
while true:
    create k clusters by assigning each
        example to closest centroid
    compute k new centroids by averaging
        examples in each cluster
    if centroids don't change:
        break

```

Fig 5 pseudocode for K means

K means is very fast and efficient because the complexity of one iteration is  $K \times N \times D$

(K= number of clusters, N=number of samples, D=time to compute the distance between two points)

As the main concept of this thesis is experimental, makes sense to use unsupervised learning to find path and structure in the data.

## Main Concept of this Thesis

The main idea is to look as intrusion detection as a second line of defence to protect IOT once an intrusion is detected by activating countermeasures to minimize damages and launch counter-attacks.

In this work is explored an Anomaly based intrusion detection system that operates at the transport-layer level on top of 6LoWPAN network.

The system that proposed does not require any particular implementation of 6LoWPAN or RPL as it uses ICMPv6 control messages defined by 6LoWPAN standard.

The high-level information logged passively for the state of the network are

- Round trip Time
- Hop Distance
- Packet Loss

In this thesis mainly 2 problems are addressed:

### **Problem 1: Characterize if the network includes at least 1 malicious node**

**Problem 1.1:** Given a set of traces collected from the network, we want to characterize if the whole network includes at least 1 malicious node.

**Problem 1.2:** Given a set of traces collected from the network, we want to characterize if the whole network includes at least 1 malicious node and also identify the type of attack.

## **Problem 2: Characterize if one node in a network is a malicious node**

**Problem 2.1:** Given a set of traces collected from the network, we want to characterize each node in the network if it is a malicious node.

**Problem 2.2:** Given a set of traces collected from the network, we want to characterize each node in the network if it is affected by a malicious node.

**Problem 2.3:** Given a set of traces collected from the network, we want to characterize each node in the network if it is a malicious node and also identify the type of attack.

**Problem 2.4:** Given a set of traces collected from the network, we want to characterize each node in the network if it is affected by a malicious node and also identify the type of attack.

# Chapter 2: Use Cases and Data Creation

In this chapter we analyze how we have created a use case to see if we can use Machine Learning Techniques to find out if there is an intrusion in the Internet of Things Network.

## Real Use Case: IOT-LAB

To create the first attempt the platform [FIT IOT-LAB](#) lab was used.

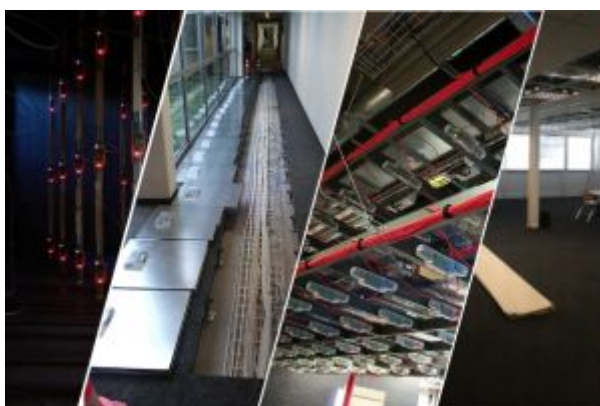
IoT-LAB provides a very large scale infrastructure suitable for testing small wireless sensor devices and heterogeneous communicating objects.



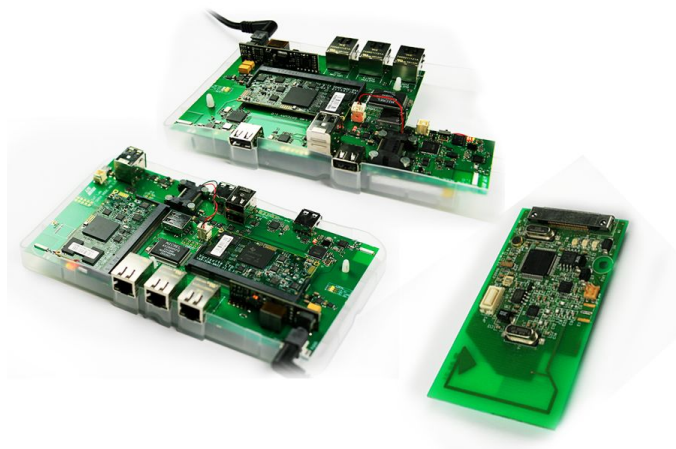
This was used to emulate a large scale network with wireless sensors node, really deployed in France, logging the communication between the real nodes.

Thanks to this use case we could work on real firmware and try real nodes and infrastructure. Although this experiment has limitations due to the fact that we don't have neither physical access to the nodes or the router neither that we have administrator privileges over the network.

IoT-LAB provides full control of network nodes and direct access to the gateways to which nodes are connected, allowing researchers to monitor nodes energy consumption and network-related metrics, e.g. end-to-end delay, throughput or overhead. The facility offers quick experiments deployment, along with easy evaluation, results collection and analysis. Defining complementary testbeds with different node types, topologies and environments allows for coverage of a wide range of real-life use-cases.



IoT-LAB testbeds are located at six different sites across France which gives forward access to 1786 wireless sensors nodes: Inria Grenoble (640), Inria Lille (293), Inria Saclay (264), ICube Strasbourg (400), Institut Mines-Télécom Paris (160) and CITI Lab Lyon (29).



## Simulated Use Case: Cooja Simulator

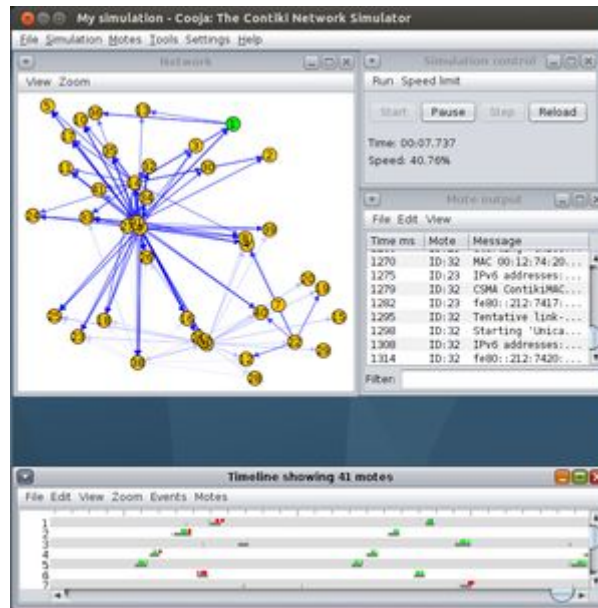


Fig 6 Example of Cooja Simulator running on Contiki Os

The second attempt was made using a platform that simulate a network of Wireless sensors. The name of the tool is [Cooja Simulator](#), part of Contiki OS.

Contiki OS is the Open Source Operating System for the Internet of Things. Contiki connects low-cost, low-power microcontrollers to the internet, making it a powerful toolbox for building a complex wireless System.

Contiki OS is can correctly simulate the behavior of a network of Internet of Things, we can use it to create different network topologies and in less time compared with the real use case, giving us the permission to create node with malicious behaviour.

This tool is also developed by a world-wide team of developers with contributions from Atmel, Cisco, ETH, Redwire LLC, SAP, Thingsquare, and many others, led by Adam Dunkels of Thingsquare.

## Contiki Features

**Memory Allocation:** Contiki is designed for tiny systems, having only a few kilobytes of memory available. Contiki is therefore highly memory efficient and provides a set of mechanisms for memory allocation: memory block allocation `memb`, a managed memory allocator `mmem`, as well as the standard C memory allocator `malloc`.

**Full IP Networking:** Contiki provides a full IP network stack, with standard IP protocols such as UDP, TCP, and HTTP, in addition to the new low-power standards like 6lowpan, RPL, and CoAP. The Contiki IPv6 stack, developed by and contributed to Contiki by Cisco, is fully certified under the IPv6 Ready Logo program.

**Power Awareness:** Contiki is designed to operate in extremely low-power systems: systems that may need to run for years on a pair of AA batteries. To assist the development of low-power systems, Contiki provides mechanisms for estimating the system power consumption and for understanding where the power was spent.

**6lowpan, RPL, CoAP:** Contiki supports the recently standardized IETF protocols for low-power IPv6 networking, including the 6lowpan adaptation layer, the RPL IPv6 multi-hop routing protocol, and the CoAP RESTful application-layer protocol.

To use Cooja Simulator you need to download Instant Contiki from [contiki-os.org](http://contiki-os.org) with Instant Contiki, a tool of 3.0 GB that is a virtual Machine created with all the necessary toolchains and software for development.

After unzipping the downloaded file, Virtual machine Manager is needed, such as Virtualbox and VMWare, for this example I decided to use Virtualbox.

In Virtualbox you need to create a new virtual Machine with the following specifications

- Type: Linux
- Version: Ubuntu 32-bit
- Memory: default
- Existing Virtual Hard Disk: Instant\_Contiki\_Ubuntu\_12.04\_32-bit.vmdk

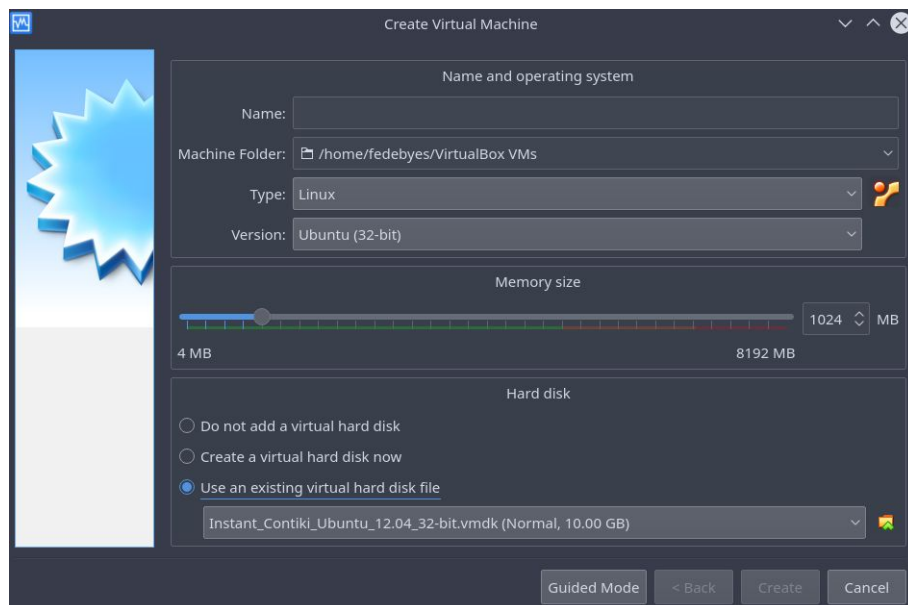


Fig 7 Settings for the virtual Machine

After that we are presented with the basic system environment

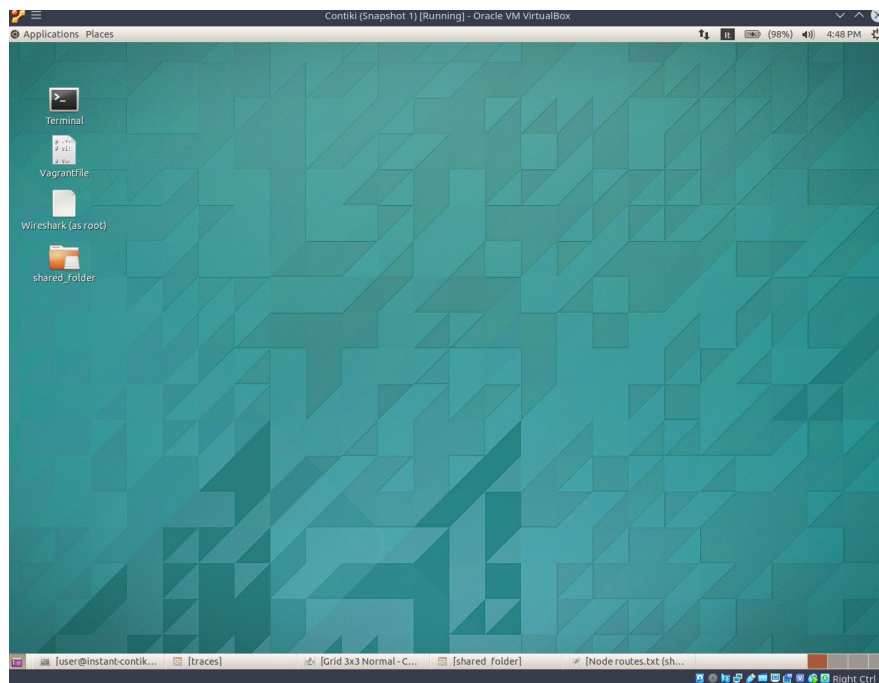


Fig 8 Environment of Instant Contiki 3.0

Here can run the Cooja Simulator opening a terminal and digiting

```
cd contiki/tools/cooja
ant run
```

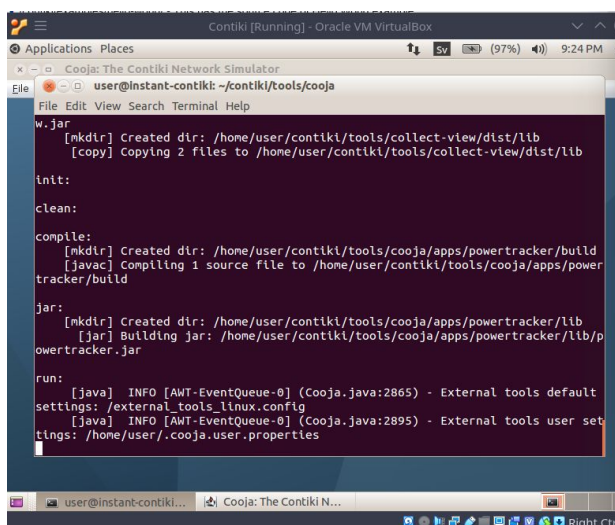


Fig 9 Contiki Simulator log

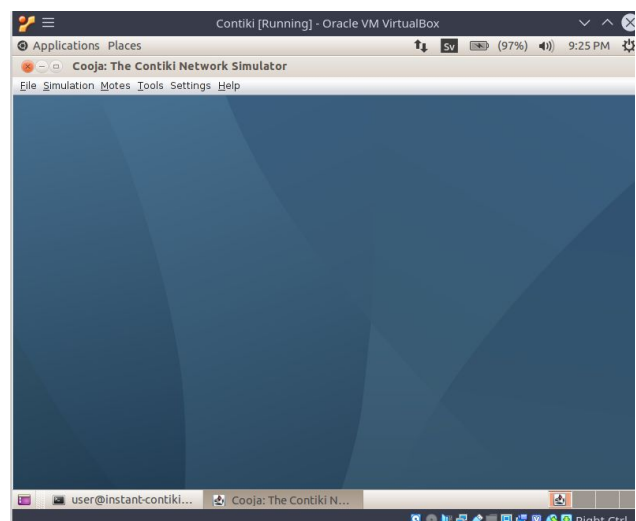


Fig 10 Contiki Simulator

Where the first simulation is created with these parameters

- Radio Medium (UDGM)

- Mote startup delay (1000)
- Random seed: 123456

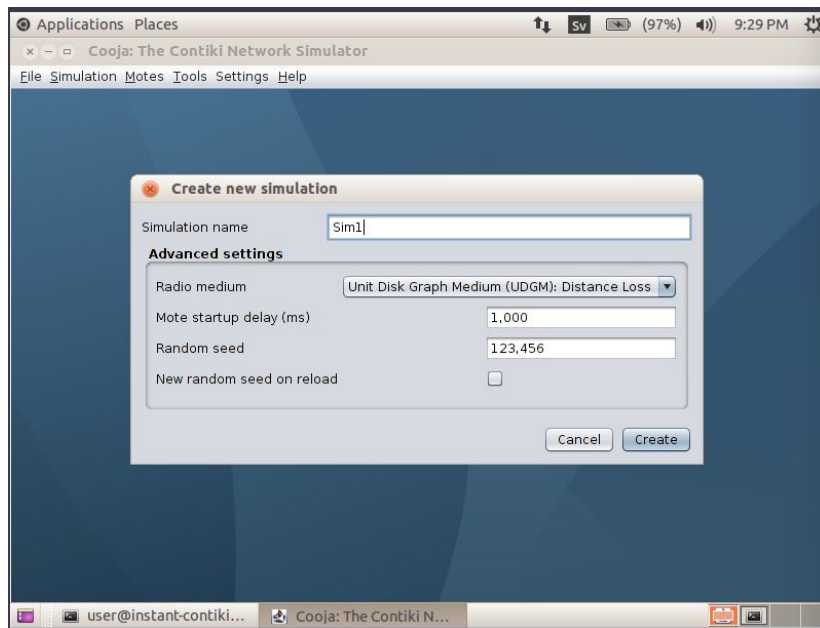


Fig 11 Simulation Settings

After that the network should be created, adding firstly the router (Sky Mote) with the firmware of the router.

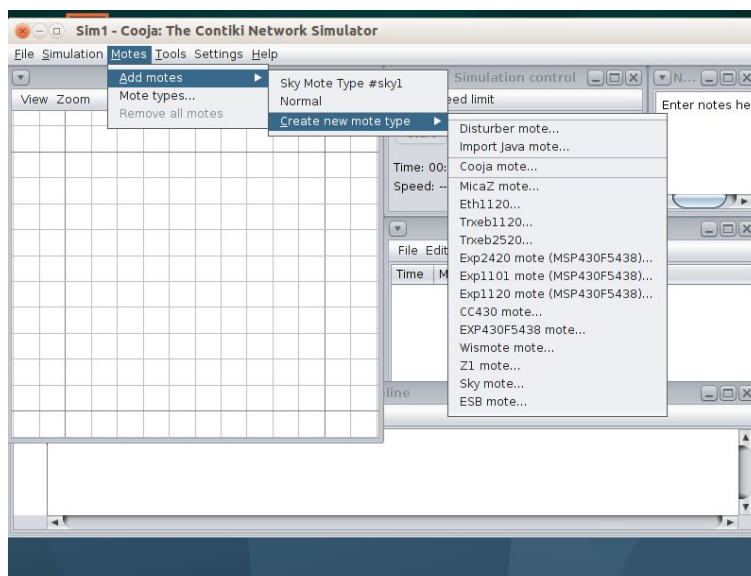


Fig 12 Adding Motes



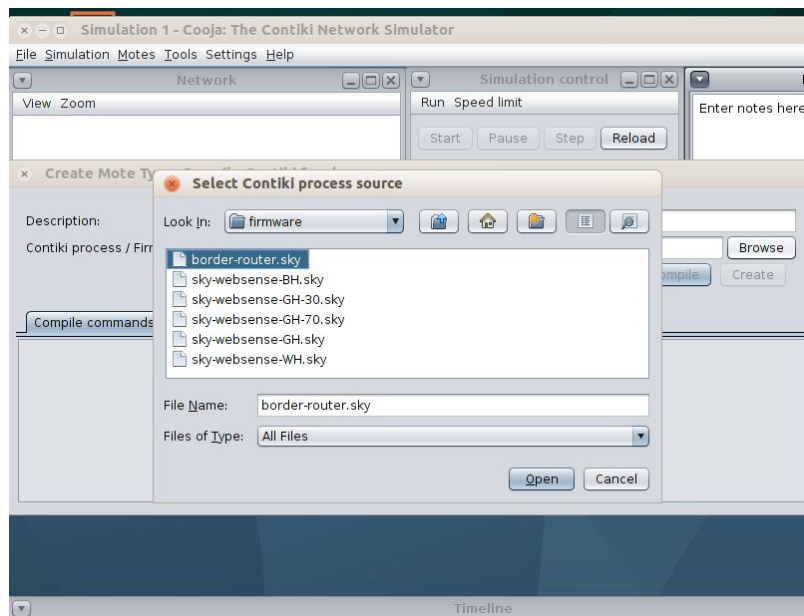


Fig 13 Selecting Firmware

There are different types of firmware:

- border-router.sky
- sky-websense-BH.sky
- sky-websense-GH.sky
- sky-websense-GH30.sky
- sky-websense-GH70.sky
- sky-websense-WH.sky

**border-router** is the firmware for the router, while the others represent the firmware for a Sky motes with different behaviour.

**WH White Hole** is the firmware for a normal behaving node.

**BH Black Hole** represent the firmware for a node that is behaving like a Black Hole, where the incoming or outgoing traffic is silently discarded or dropped, without informing that the data did not reach its intended recipient.

**GH Gray Hole** is a firmware where the node drop 50% of the packets incoming and outgoing, similar to Black Holes, there are 3 types, GH30, GH and GH70 that respectively drop 30, 50 and 70 percent of the traffic.

To obtain the network 9 more nodes are created with the firmware of the normal behaving node (sky-websense-WH.sky) and positioning them until getting a network similar to this one.

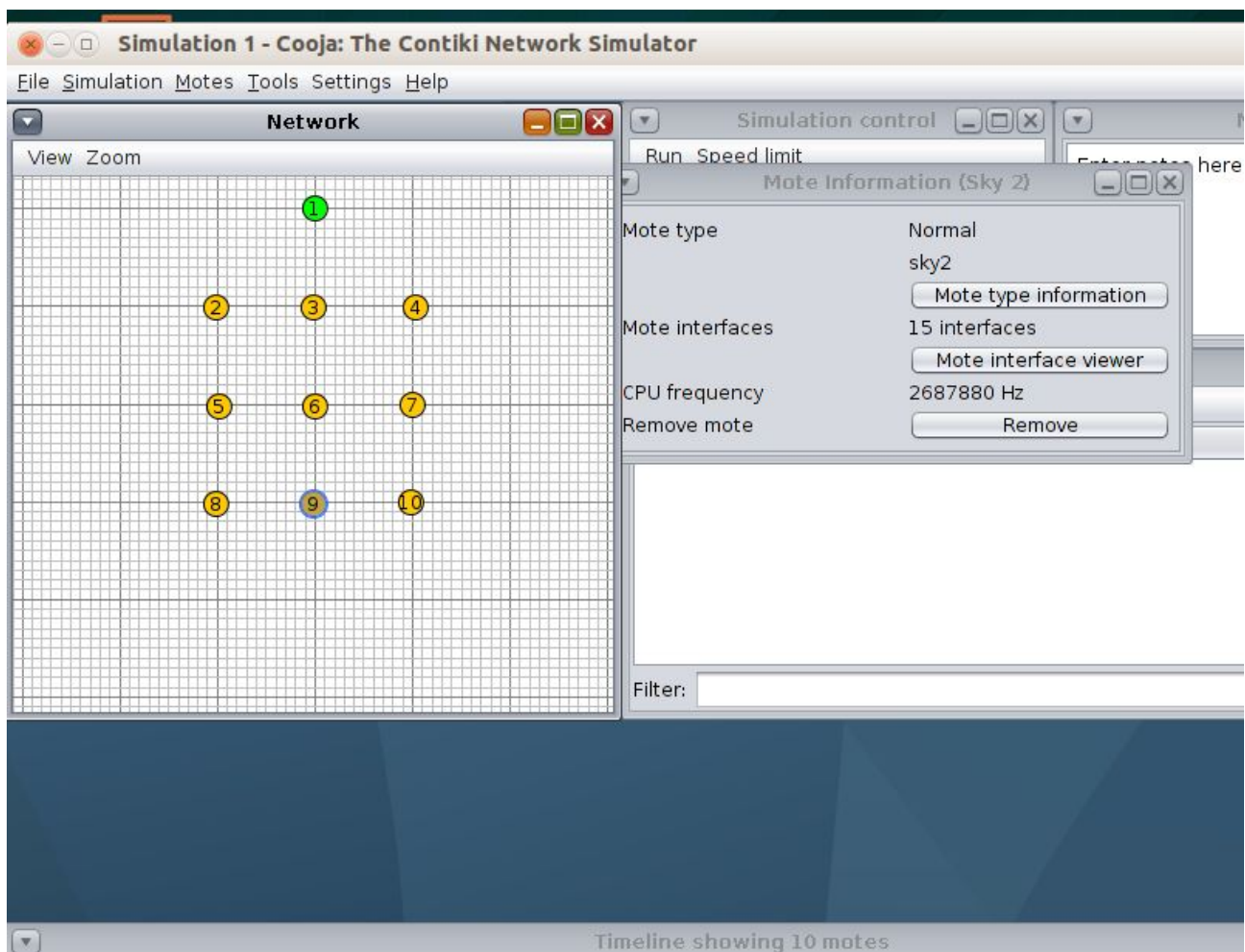


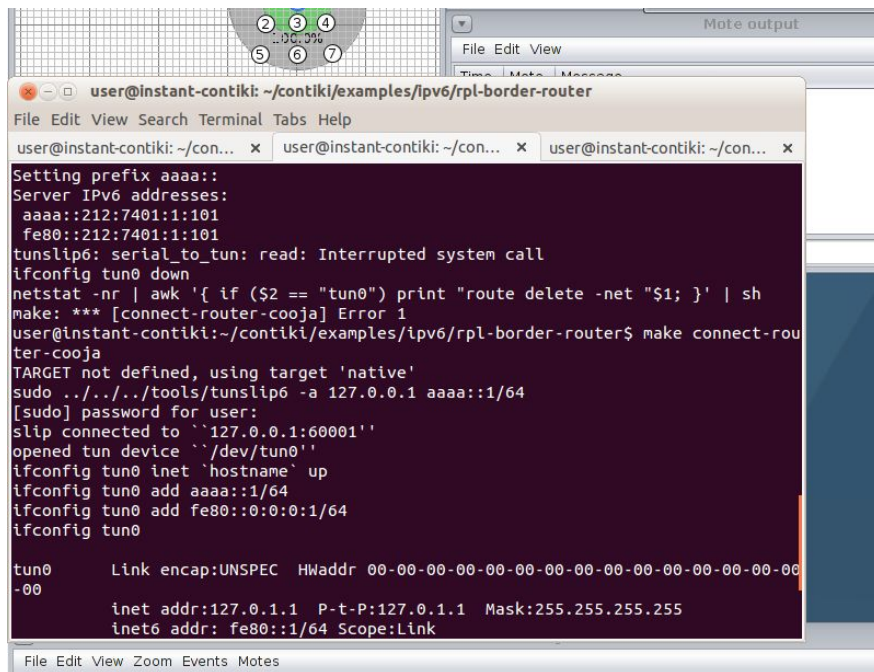
Fig 14 Network of Nodes

After that the node 1 is selected to be accessible from the network to collect data in Mote tools for Sky 1> Serial Socket (SERVER) and clicking on Start

Now the router is listening on command from the ip `127.0.0.1:60001`

To connect to the router

```
~/contiki/examples/ipv6/rpl-border-router$ make connect-router-cooja
```



The screenshot shows a terminal window titled "user@instant-contiki: ~/contiki/examples/ipv6/rpl-border-router". The terminal output includes the following commands and their results:

```
Setting prefix aaaa::
Server IPv6 addresses:
  aaaa::212:7401:1:101
  fe80::212:7401:1:101
tunslip6: serial_to_tun: read: Interrupted system call
ifconfig tun0 down
netstat -nr | awk '{ if ($2 == "tun0") print "route delete -net \"$1; }' | sh
make: *** [connect-router-cooja] Error 1
user@instant-contiki:~/contiki/examples/ipv6/rpl-border-router$ make connect-router-cooja
TARGET not defined, using target 'native'
sudo ../../tools/tunslip6 -a 127.0.0.1 aaaa::1/64
[sudo] password for user:
slip connected to `127.0.0.1:60001'
opened tun device `/dev/tun0'
ifconfig tun0 inet 'hostname' up
ifconfig tun0 add aaaa::1/64
ifconfig tun0 add fe80::0:0:0:1/64
ifconfig tun0

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
-00
          inet addr:127.0.1.1  P-t-P:127.0.1.1  Mask:255.255.255.255
          inet6 addr: fe80::1/64 Scope:Link
```

Fig 15 Terminal with the commands for router listening

If clicking on “Start” will start the simulation and the nodes communicate while the program is showing which ipv6 packets are transmitted.

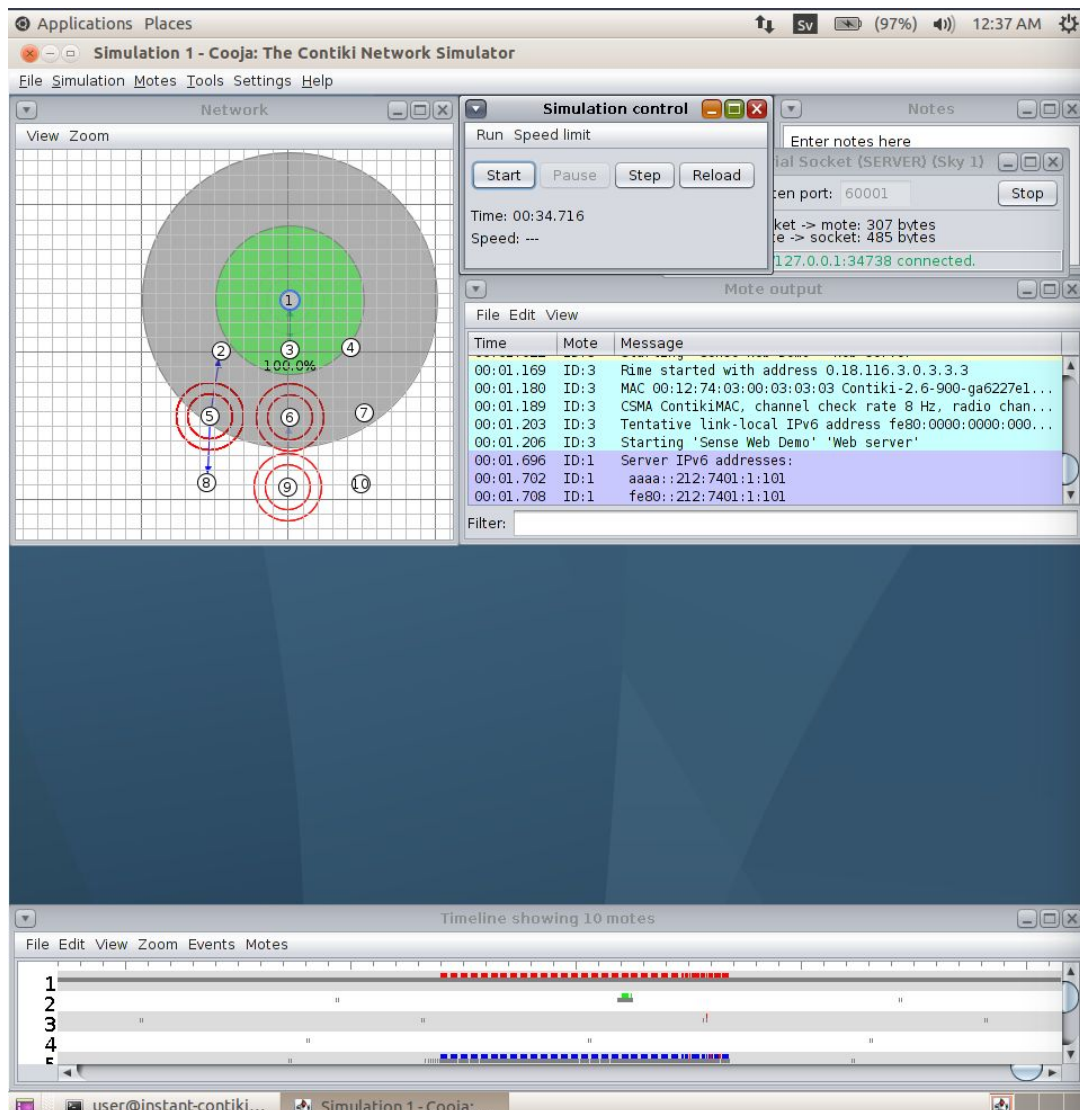


Fig 16 Network interaction

To create the use case a shell file is used to to log the information of every packet of every node of the network.

```

get-trace0.2.sh x
#!/bin/bash
type=25-normal
total=200
delay=5
date=$(date +%F_%H:%M)_
logfile=$type$date
declare -a nodes=( "aaaa::212:7402:2:202"
                    "aaaa::212:7403:3:303"
                    "aaaa::212:7404:4:404"
                    "aaaa::212:7405:5:505"
                    "aaaa::212:7406:6:606"
                    "aaaa::212:7407:7:707"
                    "aaaa::212:7408:8:808"
                    "aaaa::212:7409:9:909"
                    "aaaa::212:740a:a:a0a")
dir="./traces/$type"
mkdir $dir
echo "creating case $type"
for i in "${nodes[@]}"
do
    echo "Ping $i"

    log="$dir/$logfile$i.log"
    resp=`ping6 -c $total -i $delay $i > $log &`
    echo $resp
    sleep 1
done

log="$dir/$logfile""routes.log"
resp=`lynx -dump http://[aaaa::212:7401:1:101] > $log &`
echo $resp
msg="I'm done"
#echo $msg

```

Fig 17 Bash script for logging

This file is written in bash and define:

- **type:** type of the experiment
- **total:** total number of pings
- **delay:** delay between
- **date:** date of the experiment
- **logfile:** name of the logfile

This program declares the nodes, then creates the directory if not exists and start to ping all different nodes, every 5 seconds for 200 times, for a total of 1000 seconds or 16.6 minutes. The command ping 6 is then logged directly in a different file for each node.

So in the end of this procedure there are as many files as nodes in the network.

A ping is sent to every node and three main informations are gathered:

- ICMP sequence Number (icmp\_seq)
- Time to Live (ttl)
- Round trip Time (time)



These information are represented in plain text files

```
PING aaaa::212:740a:a:a0a(aaaa::212:740a:a:a0a) 56 data bytes
64 bytes from aaaa::212:740a:a:a0a: icmp_seq=1 ttl=60 time=439 ms
64 bytes from aaaa::212:740a:a:a0a: icmp_seq=3 ttl=60 time=375 ms
64 bytes from aaaa::212:740a:a:a0a: icmp_seq=4 ttl=60 time=232 ms
64 bytes from aaaa::212:740a:a:a0a: icmp_seq=5 ttl=60 time=266 ms
64 bytes from aaaa::212:740a:a:a0a: icmp_seq=6 ttl=60 time=249 ms
64 bytes from aaaa::212:740a:a:a0a: icmp_seq=7 ttl=60 time=467 ms
64 bytes from aaaa::212:740a:a:a0a: icmp_seq=8 ttl=60 time=289 ms
64 bytes from aaaa::212:740a:a:a0a: icmp_seq=9 ttl=60 time=304 ms
64 bytes from aaaa::212:740a:a:a0a: icmp_seq=10 ttl=60 time=214 ms
64 bytes from aaaa::212:740a:a:a0a: icmp_seq=11 ttl=60 time=386 ms
64 bytes from aaaa::212:740a:a:a0a: icmp_seq=12 ttl=60 time=337 ms
64 bytes from aaaa::212:740a:a:a0a: icmp_seq=13 ttl=60 time=505 ms
64 bytes from aaaa::212:740a:a:a0a: icmp_seq=14 ttl=60 time=473 ms
64 bytes from aaaa::212:740a:a:a0a: icmp_seq=15 ttl=60 time=243 ms
64 bytes from aaaa::212:740a:a:a0a: icmp_seq=16 ttl=60 time=431 ms
64 bytes from aaaa::212:740a:a:a0a: icmp_seq=17 ttl=60 time=388 ms
64 bytes from aaaa::212:740a:a:a0a: icmp_seq=18 ttl=60 time=760 ms
```

Fig 18 Example of log

## Data Types

The dataset that faced were made of two types:

- 9 Nodes
- 16 Nodes

25 nodes type could not be efficiently studied as a lot of wireless interferences takes place and also as the simulator need more power some of the packets are not received.

For every Dataset there were different implementation both in Random order Grid order.

### Random Grid

In this positioning the nodes are created in random position in the Network in a way that there is a path between all the nodes, so the nodes are not farther than the maximum range.

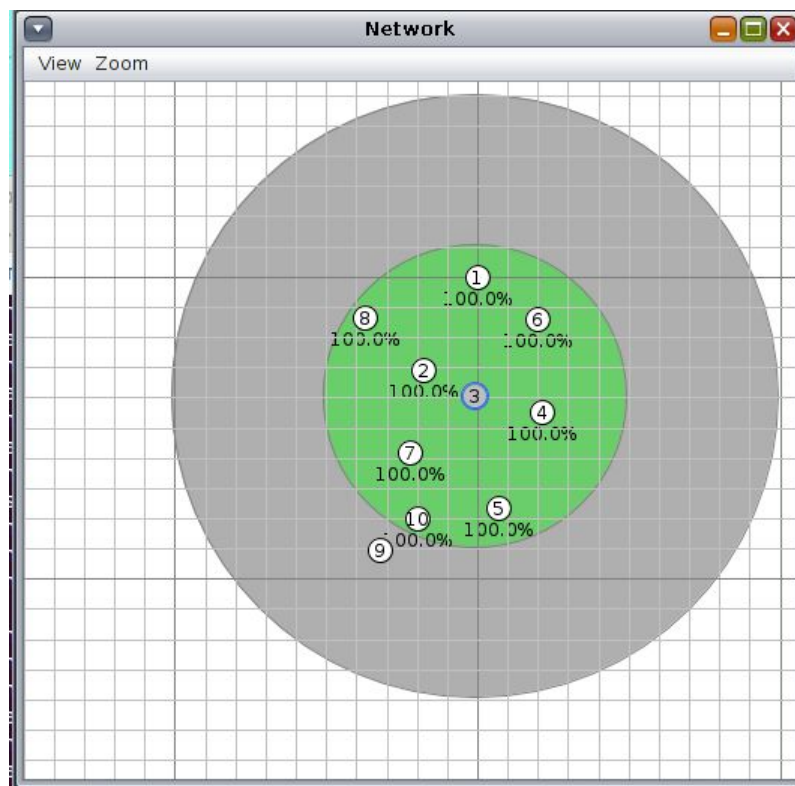


Fig 19 Example of Random Grid

### **Square grid**

The nodes are created in a way that they create a square grid with on the top the Router. In this way it is more ordered and the traffic can be measured efficiently.

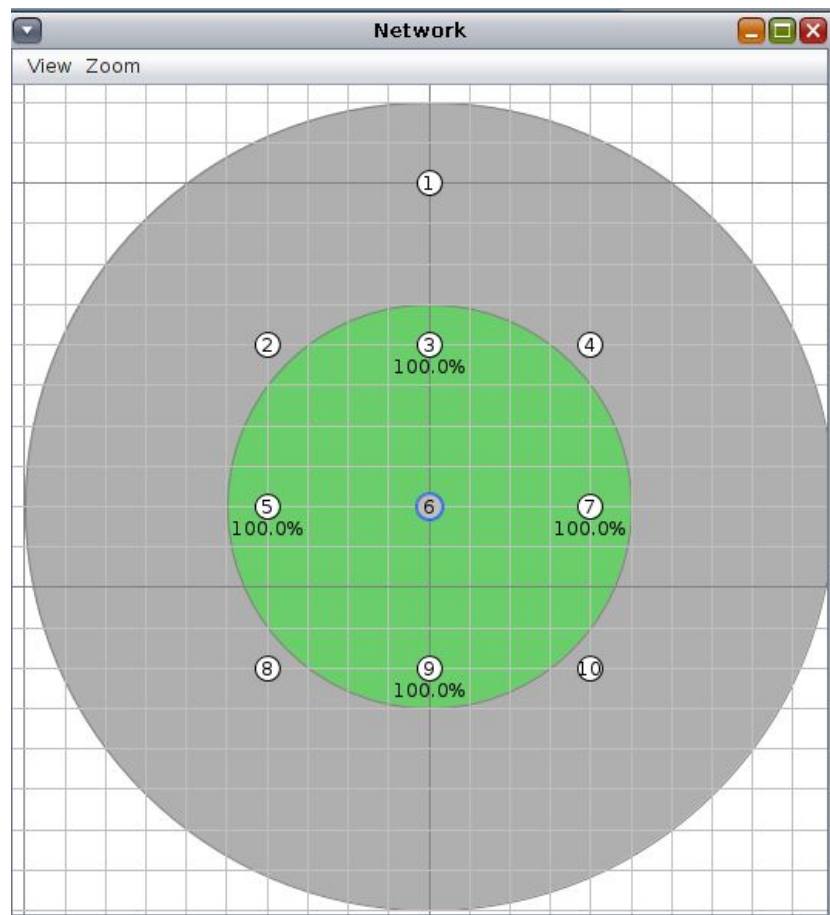


Fig 20 Example of Square Grid

### 9 Nodes

- Random Grid
  - Normal 2 Traces
  - Gray Hole 3 Traces
  - Black Hole 4 Traces
- Square Grid
  - Normal 3 Traces
  - Gray Hole 5 Traces
  - Black Hole 10 Traces

### 16 Nodes

- Square Grid
  - Normal 5 Traces
  - Gray Hole 6 Traces
  - Black Hole 2 Traces



# Chapter 3: Data analysis

## Tools

To have a comprehensive analysis of the data used:

- [Python](#) 3.7.3
- <https://jupyter.org/> 5.7.7
- PyCharm 2018.3.5

Focusing on usability and ease to use the tools that choosed are well implemented, stable and well documented.

**Python** was used mostly for calculation and the easy implementation of the Machine Learning Tools.

**Jupyter Notebook** on the other hand despite his young age, is useful due to its simplicity in running code with the interactive Python Implementation.

Having both Interactive and normal Python implementation was easier to manage using PyCharm and IDE for large Python Projects.

## Libraries

- **matplotlib** (3.0.2) Used to create plots during the analysis phase
- **json** (2.0.6) Used during the reading and editing of Json files for both Cooja and IoT labs
- **os** Integrated with the system, used to access to files and directory
- **pandas** (0.23.4) Use to manage and create DataFrame object types
- **numpy** (1.16.2) to store numbers and arrays
- **sklearn** (0.20.2) is the library with implemented all the machine learning techniques used in this thesis

## System

All the work presented in this thesis is conducted on a machine running

- Arch Linux x86\_64
- Kernel 5.0.8-arch1
- CPU Intel i7-7500U
- GPU NVidia GeForce GTX 950M

```
fedebyes@arch-laptop: ~ $ neofetch

      .o+
      000/
      +0000:
      +000000:
      -+000000+:
      /:-:++0000+:
      /++++/+++++:
      /+++++++:
      /+++000000000000/
      ./000SSSS0++0SSSSSS0+
      .0SSSSSS0-`/0SSSSSS+
      -0SSSSSS0. :SSSSSS0.
      :0SSSSSS/  0SSSS0+++
      /0SSSSSSS/  +SSSS000/-
      `0SSSSS0+/- -:/+0SSSS0+-
      +SS0+:-`    .-/+0S0:
      ++:         -/+
      .           -/+

fedebyes@arch-laptop
-----
OS: Arch Linux x86_64
Host: UX510UXK 1.0
Kernel: 5.0.8-arch1-1-ARCH
Uptime: 1 day, 19 hours, 28 mins
Packages: 2113 (pacman)
Shell: bash 5.0.3
Resolution: 1920x1080
DE: KDE
WM: KWin
Theme: Arc Dark [KDE], Breeze-Dark [GTK2/3]
Icons: breeze-dark [KDE], breeze-dark [GTK2/3]
Terminal: konsole
Terminal Font: DejaVu Sans Mono 10.5
CPU: Intel i7-7500U (4) @ 3.500GHz
GPU: NVIDIA GeForce GTX 950M
GPU: Intel HD Graphics 620
Memory: 6046MiB / 7891MiB

fedebyes@arch-laptop: ~ $ █
```

Fig 21 Host System Specifications

## Github Repository

All the code is present in the github repository <https://github.com/fedebyes/iot-netprofiler>

## Phase 1: Data Analysis per Node

As there are have different cases for the datas there was a need to create a unique way to store them.

For that reason there is a function that read all .log files from the network in every folder and put them as data easily accessible for every Case (Network disposition ei. 9 Nodes with a Black Hole in node 3 in Square grid).

So for every node there is a file .log containing the output of the command ping6 to the respective node, for every case the .log file is stored it in a different directory.



Fig 22 Content of the Log folder

For this first step was stored for every case a list of nodes that contains a custom object Node that has 3 properties:

- IP = The ID (or IP of the node)
- Hop = distance from the root
- Pkts = Data Frame Object (From the Pandas Library in Python) containing all the information of the packets in 3 columns
  - Seq =Number of sequence of the packet
  - RTT = Round Trip Time

This is done thanks to the function `import_nodes_Cooja_2` present in the project called for every group of traces from the function `import_Cooja2`

```
def import_nodes_Cooja_2(directory,tracemask,node_defaults):
```

```
    files = []
```

```
    # load all files and extract IPs of nodes
```

```
    for file in os.listdir(directory):
```

```
        try:
```

```
            if file.startswith(tracemask) and file.index("routes"):
```

```
                continue
```

```
        except:
```

```
            files.append(file)
```

This function reads the files in the directory and extract the informations

```

nodes = pd.DataFrame(columns=['node_id', 'rank'])
packets_node = {}

# Load the ICMP traces
for file in files:
    packets = pd.read_csv(directory + '/' + file,
                           sep=' |icmp_seq=|ttl=|time=',
                           na_filter=True,
                           header=None,
                           skiprows=1,
                           skipfooter=4,
                           usecols=[3, 5, 7, 9],
                           names=['node_id', 'seq', 'hop', 'rtt'],
                           engine='python').dropna().drop_duplicates()

    if len(packets) < 1:
        # Nodes affected by a black hole did not receive any packet
        node_id = file[-24:-4]
        if (node_id=="aa::212:7411:11:1111"):
            node_id="aaaa::212:7411:11:1111"
            packets = pd.DataFrame(columns=['node_id', 'seq', 'hop', 'rtt'],
                                   data=[[node_id, 1, node_defaults[node_id], 1]])

            nodes.loc[len(nodes)] = [file[-24:-4], node_defaults[node_id]]
            packets_node[file[-24:-4]] = packets

    else:
        #print("qui")
        packets['node_id'] = packets.apply(lambda row:
row['node_id'][:-1], axis=1)
        #print(packets["hop"].head())
        #print(nodes)
        #nodes.loc[len(nodes)-1] = [packets['node_id'][0], 64-packets['hop'][0]]
        #print("ciao" + str(64-packets['hop'][0]))
        #print(nodes.loc[7])
        packets = packets.sort_values(by=['node_id', 'seq'],
ascending=True, na_position='first')
        packets = packets[packets['rtt'] > 1]
        packets["hop"] = 64-packets['hop']
        packets_node[packets['node_id'][0]] = packets

nodes=nodes.sort_values(by=['rank', 'node_id'])

#transformation in node
nodeList=[]

for n in packets_node.keys():
    #print((packets_node[n]).head())
    pkts=packets_node[n].drop(["node_id", "hop"],axis=1)
    #print(pkts)
    hop=int(packets_node[n]["hop"][0])
    ip=packets_node[n]["node_id"][0]
    #print(hop)
    n=node(ip,hop,pkts)
    nodeList.append(n)

return nodeList

```

Here the ICMP traces are loaded

If there are nodes that are affected by a malicious node the log file will be almost empty

All the values of RTT, Hop, node\_id and Seq are extracted and put in a list of packets

This list is then transformed to the common interface of nodes

```
data[0][0].pkts
```

	seq	rtt
0	1	233
1	2	277
2	7	304
3	16	227
4	18	204
5	19	204
6	23	432
7	25	303
8	28	258

Fig 23 Example of Dataframe containing the data of every node

After this step was necessary to apply Machine Learning techniques to have a unique Data Frame of Data for every case, so the decision was to use some statistics of every node of the case as entry for this input DataFrame.

## Phase 2: Generating per Node Statistics

Before generating per node statistic is necessary to introduce the concept of windows.

The test conducted on Cooja consist in 200 pings answer, done every 5 seconds for a total of 1000 seconds (16 minutes) of test.

As the duration of the test is long and trying to emulate the aspect of a real network that is active most of the time, it is better to train the algorithm directly with a little part of the data.

It is certain that with a big amount of data some anomalies can be discovered, it is not certain if anomalies could be discovered in a small extract of these data.

An example to better understand this concept we can take as example the RTT Graph of the node `aaaa:212:7404:4:404` in the case of the Square grid of 9 nodes with a Black Hole in the node 5.

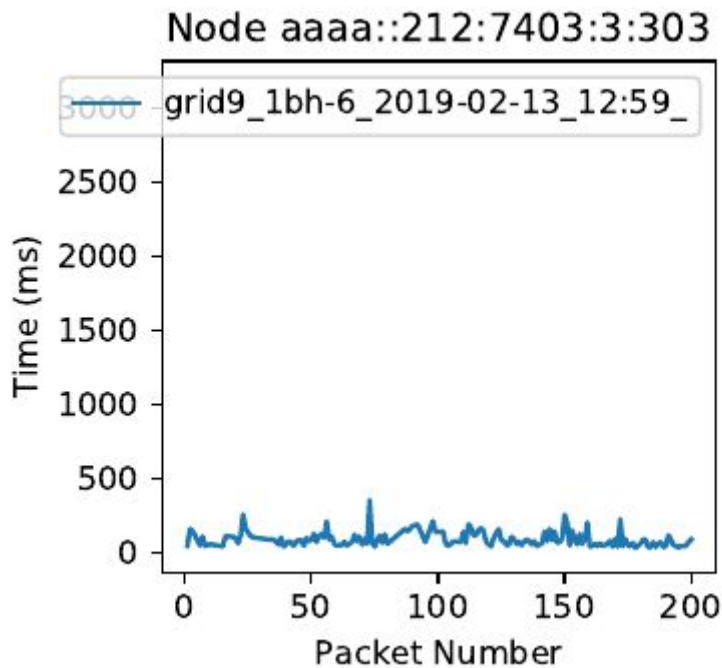


Fig 24 5 RTT Graph for node 3 in case of 1 Black Hole in node 6

We can see that we have the packet number between 0 and 200, so we can imagine to divide it in 2 windows of 100 packets or 4 windows of 50 packets.

Another idea could be to divide it of windows that partly complement each other to have virtually more data (EI Window 1: packets 0 to 50, Window 2: packets 25 to 75, Window 3: packets from 50 to 100, etc).

For now the statistics are created with 4 different sizes of windows:

- 200 packets (1 window)
- 100 packets (2 windows)
- 50 packets (4 windows)
- 25 packets (8 windows)

Plotting all graph in a big table give us the possibility to understand how the malicious node affect the other nodes.

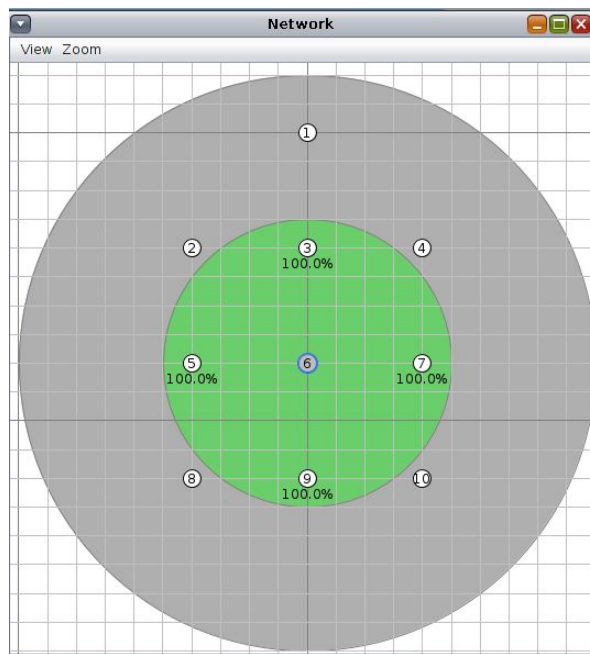


Fig 25 Example of 9 nodes network

In the case of figure 25, where node 6 is a Black Hole, watching the graph it is clear that affect just the node 9.

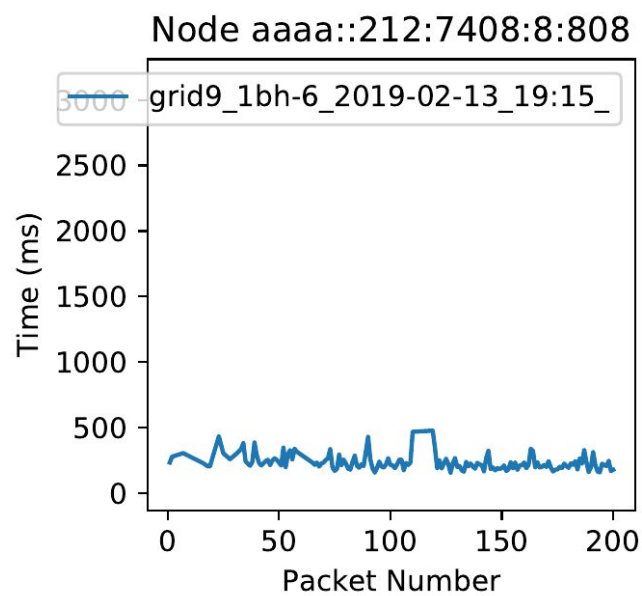


Fig 26 RTT Graph for node 8 in case of 1 Black Hole in node 6

In fact the RTT graph, with time on y axis and the sequence number on X axis show a normal behaviour with just little packet loss (seq 100-120).

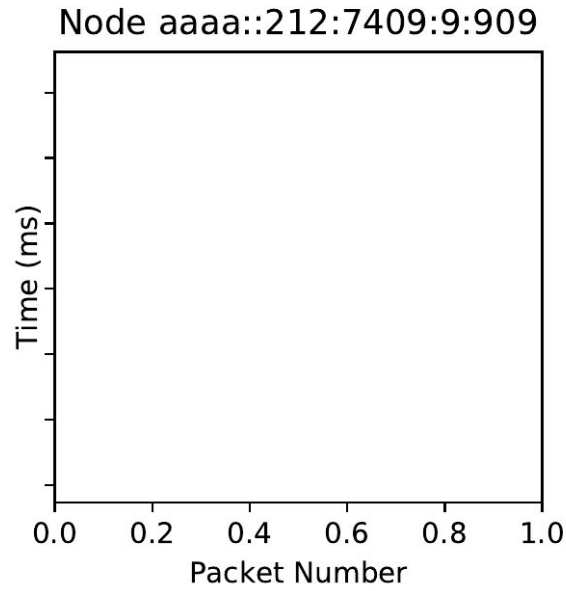


Fig 27 RTT Graph for node 9 in case of 1 Black Hole in node 6

In the RTT graph of Node number 9 (directly one hop after the Black Hole in node 6) we can see that there is total packet loss and no packet is received.

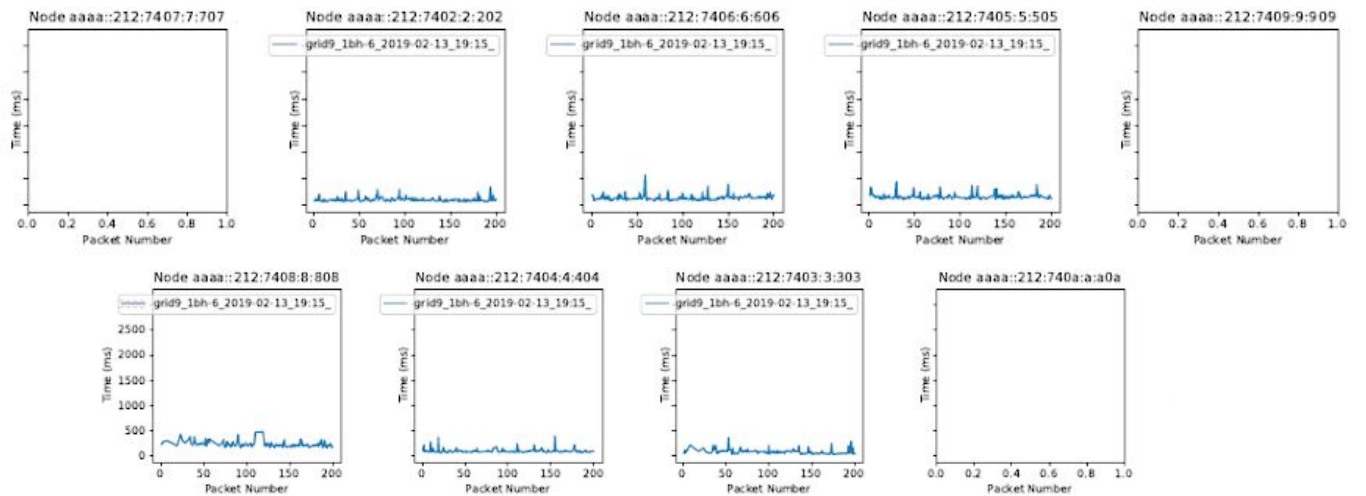


Fig 28 RTT Graph of the case grid9\_1bh6

A better overview of the figure 27 is given by all the RTT graphs in figure 28, showing that the Black Hole is receiving all the packets but not transmitting to node 9, 7 or 10.



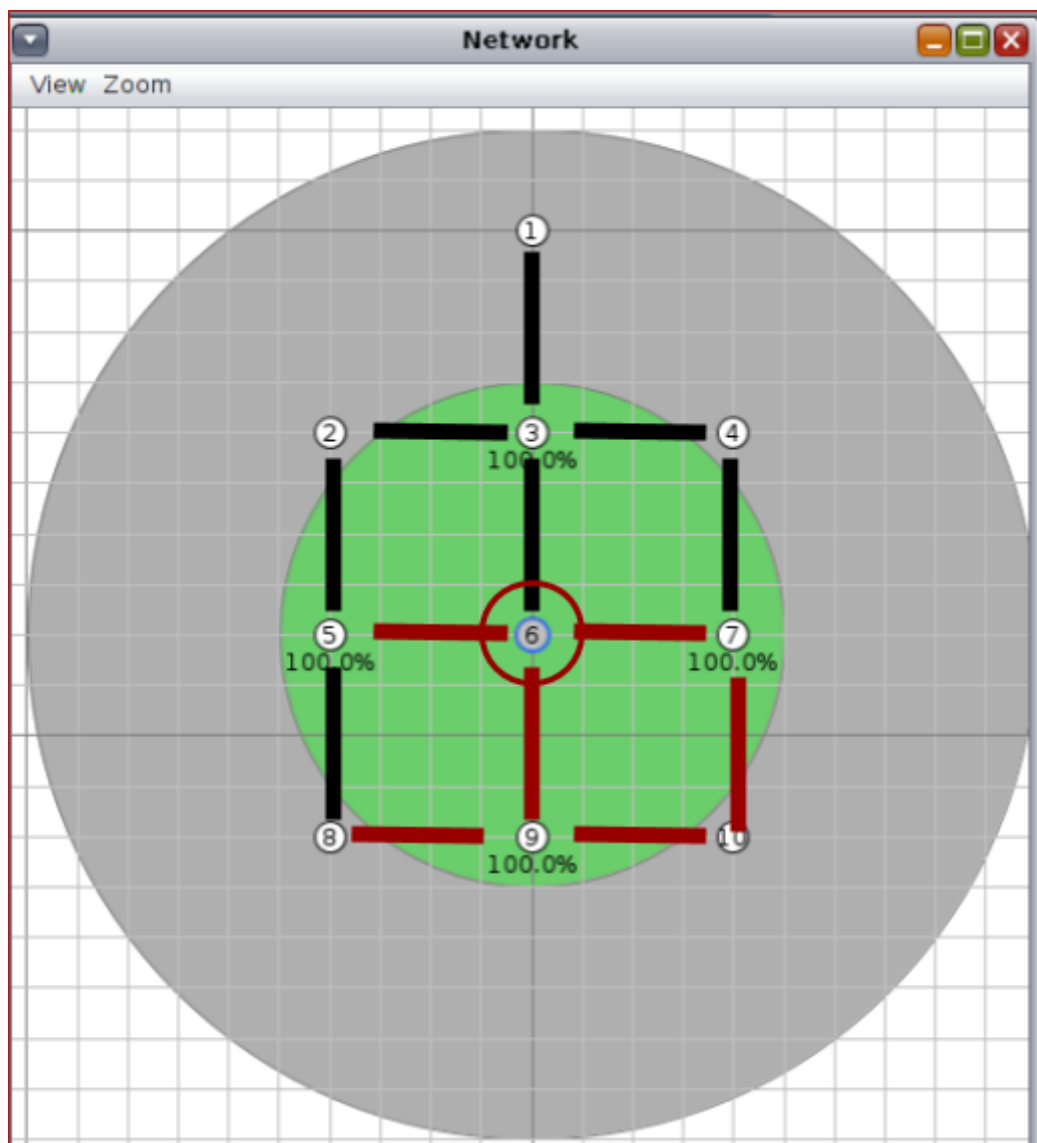


Fig 29 Links in the case of 9 nodes Square Grid

The example is clearer in the figure 37, where the Black hole is the node 6. Links between nodes are represented from a line, black in normal case, red in the case of the link is affected from the malicious node 6.

As we can see the nodes 7, 9 and 10, in this particular case node 5 and 8 are not affected because the RPL algorithm detect a different path (1->3->5->8).

[In the repository](#) is possible to find the complete pdf and study all the cases.

Other statistics analyze were the density of packets,

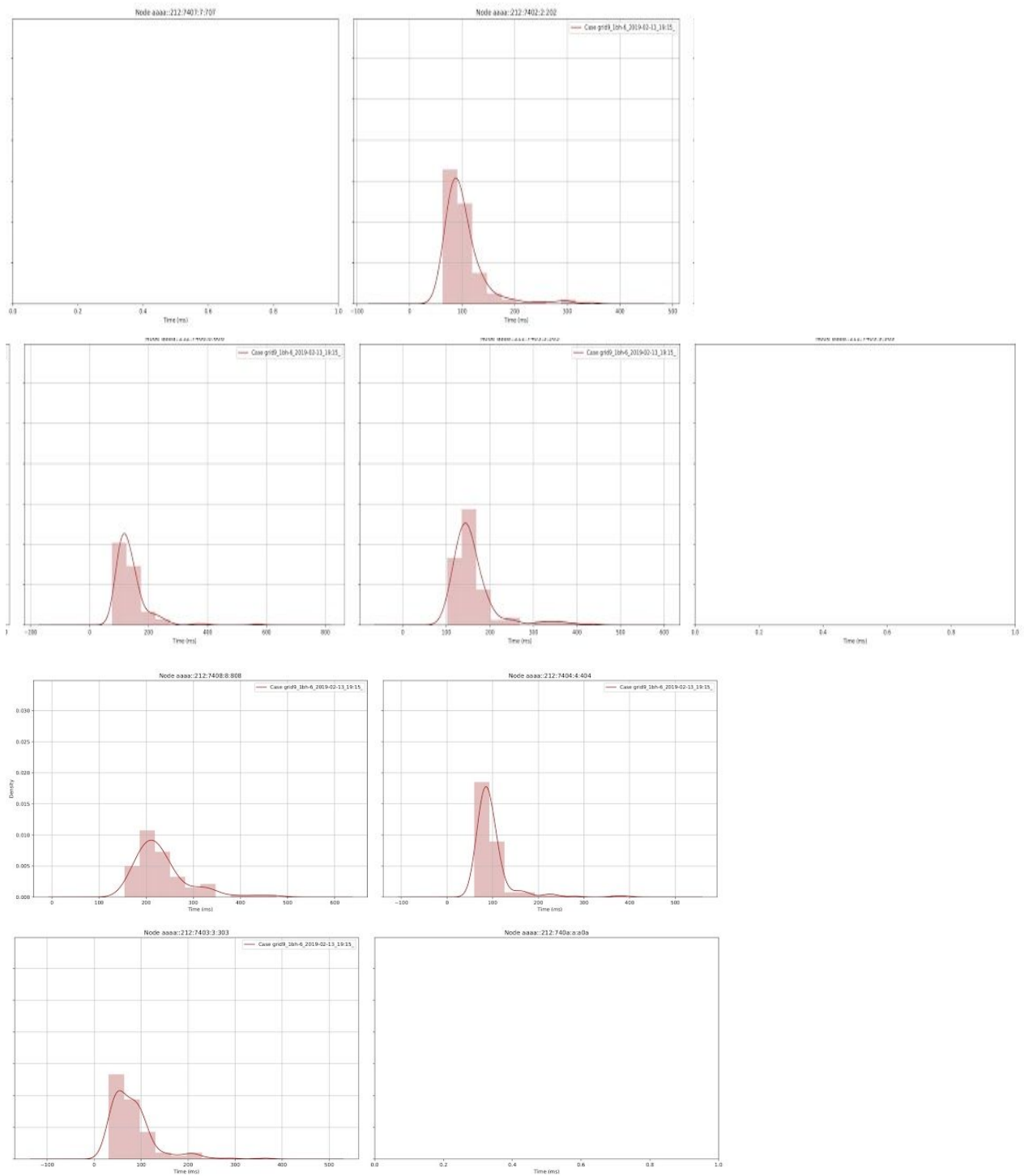


Fig 30 Density of packets for every node in the case grid9\_1bh6

This plot, made for every node in every case represent the Kernel density Estimation of the RTT of every packet.

The X axis is time in ms, while on the y axis is the density.

As expected the nodes that are far from the router have a more distributed curve, due to the different latency of every hop.

On the other hand nodes that are affected from the Black Hole don't have any packet and so no Kernel Density Estimation is possible.

In figure 30 is represented also an histogram to see how many packets have the same RTT time, expressed in milliseconds.

Another study done is the Density of delay by case

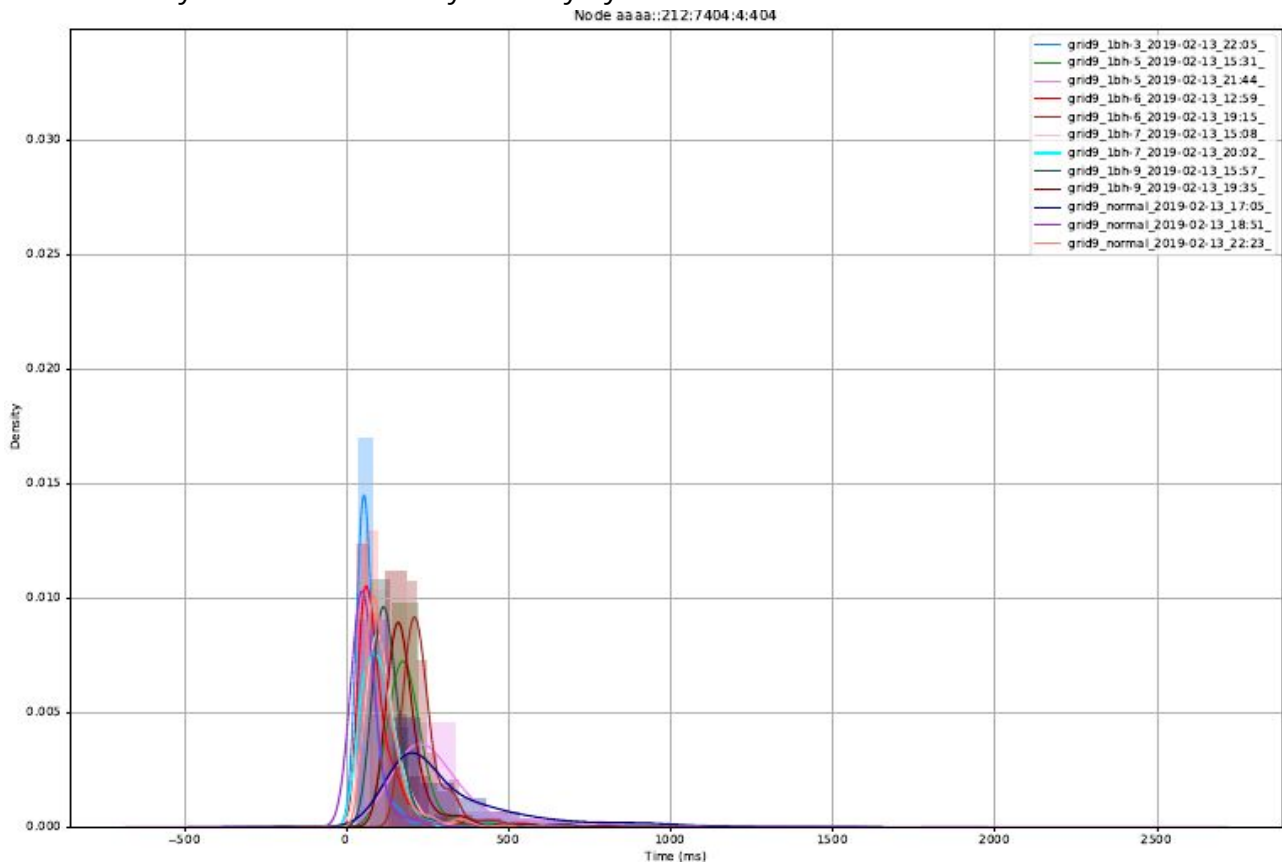


Fig 35 An extract of density of delay by case fo the node number 4

The analysis of the density of delay is done for every node as part of research process.

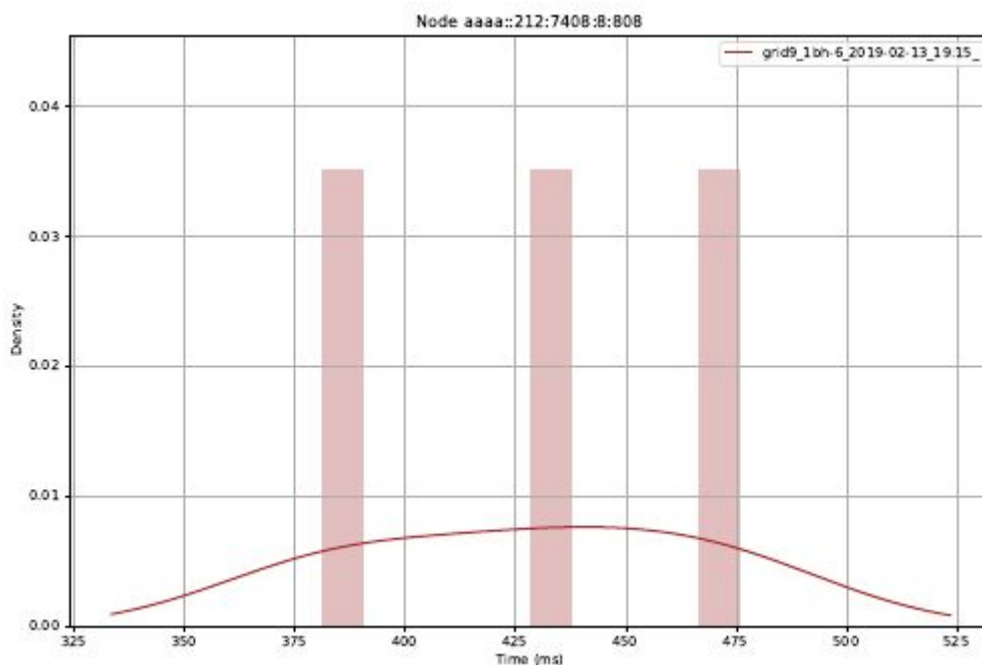


Fig 36 Analysis of density of outliers for node 8

Another research done was the density of outliers described in figure 36, this plot is generated for every node for every case. The presence of outliers is connected with the fact that a node received the packets, as if there are no packets there are not outliers, for that reason the number of outliers is included.

Thanks to the function `create_stats` this is possible and the statistics for windows are saved in a csv file to avoid recalculate every time the statistic.

```
def create_stats(directory, df, pings, window):
    cases = []
    casesAccuracy = df["case_accuracy"].values
    casesAccuracy2 = df["case_accuracy2"].values

    cases = df["case"].values
    folder = df["directory"].values + directory

    data = import_Cooja2(df, directory)
```

This function creates the statistic for every node in the window

```
print("Processing...")
d = {"experiment": [],
     "node_id": [],
     "label": [],
     "label_2": [],
     "loss": [],
     "count": [],
     "std": [],
     "mean": [],
     "var": [],
     "hop": [],
     "min": [],
     "max": [],
     "outliers": []}
```

The features of the dataframe that we want to create are defined in the dictionary `d`

```

        "window": []
    }
    # count=[]
    labels = []
    var = []
    n = pings

    for i in range(len(data)):
        # window=pings[i]

        for j in range(len(data[i])):

            for z in range(0, n, int(window)):
                node = data[i][j].pkts

                name = str(j) + " " + cases[i]
                nodeWindow = node[(node["seq"] < z + window) &
(node["seq"] >= z)]
                nodeWindowP = nodeWindow["rtt"]
                d["count"].append(nodeWindowP.count())
                # Case without outliers
                # Case with outliers
                std = 0
                if (nodeWindowP.std() > 10):
                    std = 1
                    std = nodeWindowP.std()

                d["std"].append(std)
                mean=0

                if (nodeWindowP.mean()>mean): mean=nodeWindowP.mean()
                # if(mean<1):print(mean)
                d["mean"].append(mean)
                var = 0
                if (nodeWindowP.var() > var): var = nodeWindowP.var()
                d["var"].append(var)
                d["experiment"].append(cases[i])
                d["hop"].append(data[i][j].hop)

                if(casesAccuracy[i]=="normal"):
                    d["label"].append("Normal")

                else:
                    d["label"].append("Attacked")

                if (casesAccuracy[i] == "normal"):
                    d["label_2"].append("Normal")
                elif(casesAccuracy[i] == "BH"):
                    d["label_2"].append("BH")
                else:
                    d["label_2"].append("GH")
                d["outliers"].append(getOutliers(nodeWindow)["rtt"].count())
                missing = window - nodeWindow.count()
                d["node_id"].append(data[i][j].ip)
                mP = getPercentageMissingPackets(nodeWindow, window)
                d["min"].append(data[i][j].pkts["rtt"].min())
                d["max"].append(data[i][j].pkts["rtt"].max())
                d["loss"].append(mP)
                d["window"].append(window)

    stats = pd.DataFrame(d)
    stats.to_csv(directory + "stats.csv", sep=',', encoding='utf-8')
    return stats

```

With 3 for loops we check the type of traces, then we extract the node statistic and with the third the statistic is generated per window

Generation of sequence value

Generation of std value

Generation of mean value

Generation of var, experiment and hop value

The label is given here from the name of the network

Generation of outliers, node\_id, min, max, loss and window value.

The statistics created are of course different for every window and for this experiment we choose to calculate the following statistics:

- loss: Packet loss for the window
- count: Number of packets received per window
- std: Standard deviation of the RTT time of the packets
- mean: Mean of the RTT time of the packets
- var: Variance of the RTT time of the packets
- hop: hop of the node when the packet was received
- min: minimum RTT time of the packets
- max: maximum RTT time of the packets
- outliers: number of packets that have a value of RTT very far from the average RTT time
- window: amplitude of the window

So after running the function `create_stats` we have as output a Data Frame containing all the information as well as other extra information such as the name of the node and if the node was a Normal behaving Node, a Black Hole behaving Node or a Grey Hole behaving node and if the node was attacked or not.

The main problem that was faced was the mislabeling of the nodes: the nodes were labeled following the network, so if the network contained a malicious node all the nodes of the network were labeled as malicious, but some of the nodes not affected or not malicious would be then mislabeled.

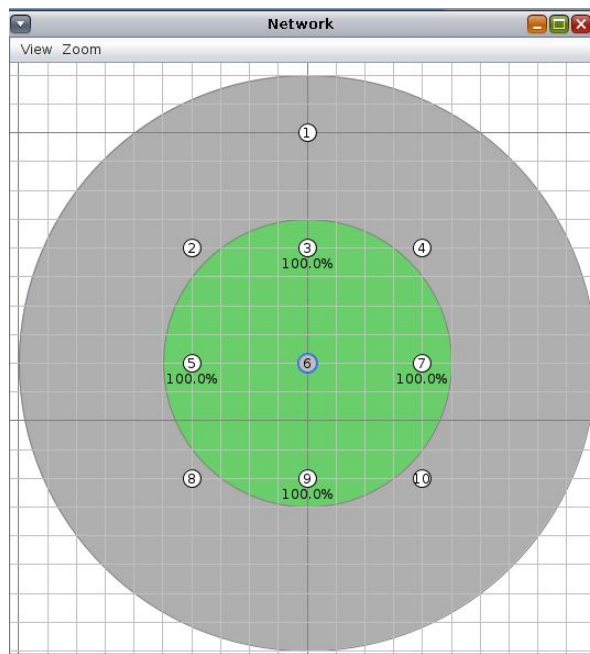


Fig 31 Example of 9 nodes network

For example in the figure 31, if the node 5 is a Black Hole, it doesn't affect nodes 10, 4, 3 and 2, so they should be labeled as normal.

To do so was necessary manual labeling after checking every case we created.

This is very important for the next chapter talking about labels and accuracy.

## Phase 3: Machine Learning Techniques

The most common Machine Learning method for Anomaly Detection is K-Means, as it is very easy to use it for any type of grouping. [48]

Kmeans is an Unsupervised learning Algorithm, that means that the computer is trained with unlabeled data and the output is not categorizable.

This is very useful to detect patterns and get meaningful insights when the human expert doesn't know what to look in the data.

At first we divide the input stats for window

```
directory="../../cooja3-9nodes/"
df = pd.read_csv(directory + "stats_per_node.csv", sep=',', encoding='utf-8').drop(columns="Unnamed: 0")

win_25_stats = df[df["window"] == 25]
win_50_stats = df[df["window"] == 50]
win_100_stats = df[df["window"] == 100]
win_200_stats = df[df["window"] == 200]
trace_stats = {
    25: win_25_stats.drop(columns=["label_2"]),
    50: win_50_stats.drop(columns=["label_2"]),
    100: win_100_stats.drop(columns=["label_2"]),
    200: win_200_stats.drop(columns=["label_2"]),
}
```

In this way we have 4 different dataset, every dataset has different data, as the statistic are calculated by window.

```
features_to_drop = [
    'node_id', 'experiment', 'label', 'window',
    'mean',
    #'loss',
    'count',
    'outliers',
    'std',
    #'var',
    'hop',
    #'min',
    'max'
]
```

For the experiment we saw that in the case of K-Means the best feature that we wanted to use where the packet loss, the variance and the minimum.

That is logically explained by the fact that as construction behavior the malicious node are having an effect of dropping a lot of packets, having the packet loss near 100%.

The core of the algorithm is in the following code



<pre> figsize= figsize=(10,30) fig, axs= plt.subplots(len(trace_stats),1, figsize=figsize,sharey=True, ) count=-1 for trace_size in trace_stats:     count+=1      trace = trace_stats[trace_size]      target = trace["label"].values      correction = []     for i in range(len(target)):         if (i == "Normal"):             correction.append(0)         else:             correction.append(1)     #dropping features     features = trace.drop(columns=features_to_drop)     kmeans = KMeans(n_clusters=2)     kmeans.fit(features)     labels = kmeans.predict(features)      #labels = f.accuracy_score_corrected(correction, labels)      trace["KmeansLabels"]=labels     centroids = kmeans.cluster_centers_     #print(trace)     #print(centroids)     #End of algorithm     #Try to draw them     X=features     pca = PCA(n_components=2)     pca.fit(X)     X_ = pca.transform(X)     dfPCA = pd.DataFrame({'x1': X_[:,0], 'x2': X_[:,1]})     dfPCA["labels"]=labels     labels = trace['KmeansLabels'].unique().tolist()     #plt.figure(figsize=(7,5))      for lab in labels:         #print(lab)          #print(dfPCA[dfPCA['labels'] == lab])         axs[count].scatter(dfPCA.loc[dfPCA['labels'] == lab, 'x1'], dfPCA.loc[dfPCA['labels'] == lab, 'x2'], label=lab)         axs[count].legend() </pre>	<p>Empty plots are created</p> <p>Every experiment (trace) is evaluated</p> <p>The correction array is created</p> <p>Features are dropped K means algorithm from skitlearn is called with 2 clusters The prediction is assigned to the label array</p> <p>the centroids of the cluster are saved</p> <p>PCA algorithm is called to reduce the number of dimension to 2 to plot the results and understand better how kmeans clusterized the results</p> <p>The results are plotted in the plots and colored by cluster of pertinency</p>
--	---

At first we can use just 2 cluster to identify if a node is malicious or not, then we can try to use 3 to identify if also a node is a Gray Hole behaving.

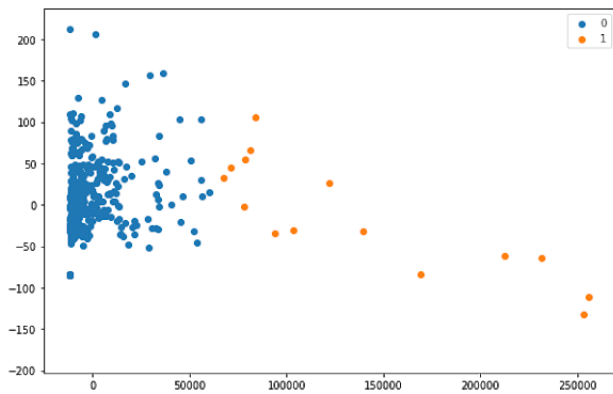
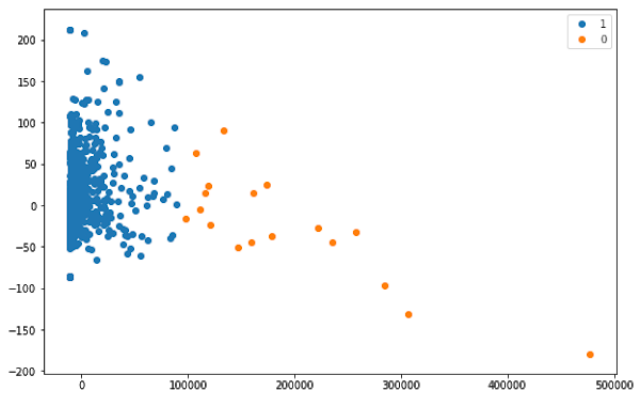


Fig 32 Cluster division (after PCA) of the nodes label with windows of 25 and 50

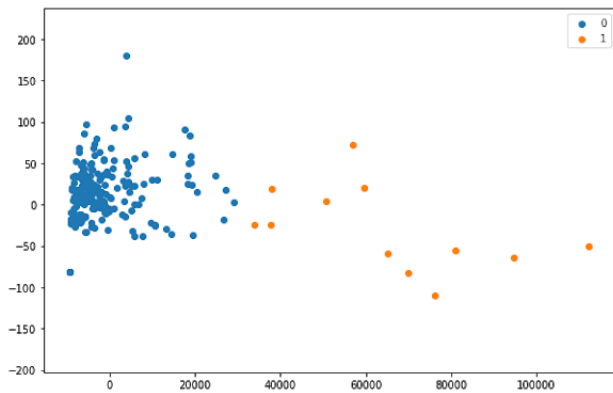
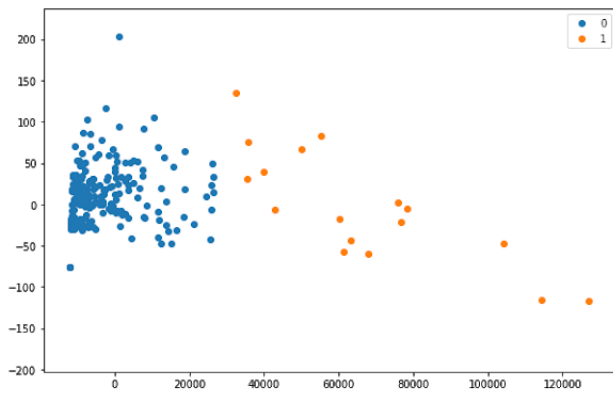


Fig 33 Cluster division (after PCA) of the nodes label with windows of 100 and 200

In this case we can easily see how K-Means divide in two different clusters the example with 9 nodes, in all different cases pointing out in orange all the misbehaving nodes. The 4 graphs are the 4 different sizes of windows (25, 50, 100 and 200).

In this case the code is runned using PCA (from sklearn) a way to reduce the dimensions of the cluster to better understand how the Algorithm divided the results using just 3 parameters (loss, var and min)

This result because we can easily see that there are more normal behaving nodes and the number of misbehaving nodes match the number of nodes inserted in the differents experiments.

In this case we saw that for example we can easily detect nodes that are not receiving packets but we cannot detect the node that is producing the problems. In other words we can detect the effect of the problem but not necessarily the problem.

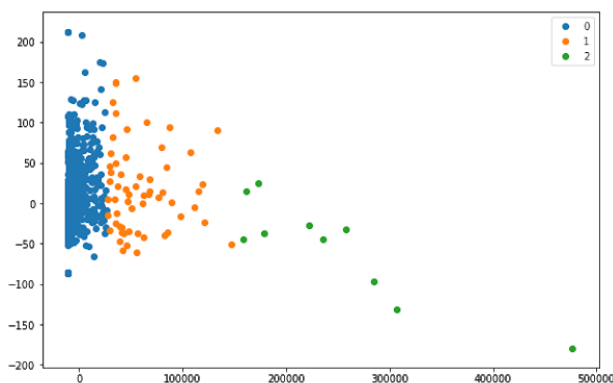


Figure 28 Cluster division (after PCA) of the nodes label using 3 clusters

In the same way we can use the algorithm to create more cluster and check the effect of different types of attacks (Black Holes and Gray Holes)

In the figure 28 we have the same example of the previous image (9 Nodes with windows of 25) and the same parameters with just one change: the output of the K-Means algorithm is 3 Cluster and not 2.

Here there is an extract of the code that runs the experiment with K-Means and the 3 clusters, thanks to skitlearn the implementation of K-Means is very easy and intuitive.

## Phase 4: Accuracy of the method

This accuracy is for predicting with K-Means if a node was Malicious or not is high (76.8%) for every type of window (25, 50, 100, 200).

Due to the limitation of unsupervised learning it is not possible to calculate the accuracy for the prediction of the three cases attack, as the cluster are assigned randomly from the machine.

I strongly suggest to use Supervised Learning in this case study.

In a more production wise tool probably the best Idea would be to use Unsupervised learning In a big industry is impossible to create entry and labels for attacks to train the machine learning algorithm.

Other solution elaborated implement the use of supervised learning algorithm such as KNN, Random Forest and SVM, these solution are very powerful and give us pretty good results in this experiment even with the separate labeling in 3 classes.

The separate labeling requires a lot of time because the expert need to check every layout of the network checking every node statistics.

Model	Window Size	Mean Accuracy
Random Forest	25	0.843773
Random Forest	50	0.922656
Random Forest	100	0.990122
Random Forest	200	0.848522
KNN	25	0.823385
KNN	50	0.842962
KNN	100	0.816435
KNN	200	0.816435
SVM	25	0.847797
SVM	50	0.879224
SVM	100	0.845389
SVM	200	0.845389

Fig 34 Results of different Algorithm saved in a dataframe

The solution of using supervised learning, as previously stated, has the big limits to being linked to the fact that a human expert need to manually label the data, like in this case.

This method peaked the 99% percent of right prediction, but this kind of methods is very difficult to implement in a real Internet of Things Network such and Industry.

# Chapter 4: Conclusions and Future Work

As stated in the previous chapter this method is very efficient and useful to understand if there is an Intrusion in the network.

Let's see in conclusion the solution to the different problems proposed in the first Chapter

**Problem 1.1:** Given a set of traces collected from the network, we want to characterize if the whole network includes at least 1 malicious node.

**Solution 1.1:** This problem is solved using the Machine Learning technique, with both supervised and unsupervised types we can easily spot misbehaving nodes, so if in a network there is more than one misbehaving node, we can safely assume that the whole network is malicious.

**Problem 1.2:** Given a set of traces collected from the network, we want to characterize if the whole network includes at least 1 malicious node and also identify the type of attack.

**Solution 1.2:** As the previous problem we can consider it solved and with very high accuracy we can also spot the type of the attack

**Problem 2.1:** Given a set of traces collected from the network, we want to characterize each node in the network if it is a malicious node.

**Solution 2.1:** This problem is not possible to solve just observing the traces of the network, by definition a node that is malicious (a Black Hole) is correctly answering to pings but is not forwarding messages, that means for the traces we can collect there is not packet loss nor a big variance

**Problem 2.2:** Given a set of traces collected from the network, we want to characterize each node in the network if it is affected by a malicious node.

**Solution 2.2:** The solution of this problem is found thanks to the per node analysis method developed in this thesis.

**Problem 2.3:** Given a set of traces collected from the network, we want to characterize each node in the network is a malicious node and also identify the type of attack.

**Solution 2.4:** as the problem 2.1, just using the traces is not possible to divide a malicious node by a normal one because both are correctly answering to the pings

**Problem 2.4:** Given a set of traces collected from the network, we want to characterize each node in the network if it is affected by a malicious node and also identify the type of attack.

**Solution 2.3:** We saw not only that we can identify if the node is malicious, just using two cluster but even the type of attack, using three clusters.

## Conclusions

The method explored in this thesis can be easily implemented in a production environment, using a PC connected to the network where all the nodes are, or simply and more security wise running a script continuously from the router itself.

The Python function can produce datas and results more and more precise while data are gathering, so we can just let a file open for writing from one side (the stats\_per\_node.csv file) and reading from the other side.

This methods does not take into account possible fail of the network, possibility that simply the nodes are not connected anymore and limitation of the IIOT network implementation.

Using ICMP messages can give us insight about the state of the node itself but in my opinion can't give us an idea of the fact that a network is attacked, due to the limitation itself of the environment, while we can actually implement ways that are more intrusive and requires direct connection to the router to check the messages transmitted in the network.

This anomaly detection Systems is a second line of defence towards intrusion detection, when the attack already has been successful, so for that reason is a suggested system to implement but just if paired with other system to prevent the attack in the first place.

## Future work

Although the Anomaly Detection System demonstrated to resolve the problems towards some attack such as Black Holes and Grey Holes Attacks, we need to create more experiment to see if it will detect even more types of attacks. Other than that probably a more intrusive way to have insight and use the data is way more efficient, but to do so we will need to have administrator privileges to the network.

To conclude the future of the idea behind this thesis is promising and can solve one of the most important challenges that we are facing right now in the Industrial Internet of Things using tools such as Machine Learning.



# References

1. Ioannis Chatzigiannakis, Vasiliki Liagkou, Paul G. Spirakis: Brief Announcement: Providing End-to-End Secure Communication in Low-Power Wide Area Networks. CSCML 2018: 101-104
2. Dimitrios Amaxilatis, Orestis Akrivopoulos, Georgios Mylonas, Ioannis Chatzigiannakis: An IoT-Based Solution for Monitoring a Fleet of Educational Buildings Focusing on Energy Efficiency. Sensors 17(10): 2296 (2017)
3. Ioannis Chatzigiannakis, Andrea Vitaletti, Apostolos Pyrgelis: A privacy-preserving smart parking system using an IoT elliptic curve based security platform. Computer Communications 89-90: 165-177 (2016)
4. Khalil Massri, Andrea Vitaletti, Alessandro Vernata, Ioannis Chatzigiannakis: Routing Protocols for Delay Tolerant Networks: A Reference Architecture and a Thorough Quantitative Evaluation. J. Sensor and Actuator Networks 5(2): 6 (2016)
5. Geoff Coulson, Barry Porter, Ioannis Chatzigiannakis, Christos Koninis, Stefan Fischer, Dennis Pfisterer, Daniel Bimschas, Torsten Braun, Philipp Hurni, Markus Anwander, Gerald Wagenknecht, Sándor P. Fekete, Alexander Kröller, Tobias Baumgartner: Flexible experimentation in wireless sensor networks. Commun. ACM 55(1): 82-90 (2012)
6. Ioannis Chatzigiannakis, Georgios Mylonas, Andrea Vitaletti: Urban pervasive applications: Challenges, scenarios and case studies. Computer Science Review 5(1): 103-118 (2011)
7. Ioannis Chatzigiannakis, Georgios Mylonas, Sotiris E. Nikolettseas: The Design of an Environment for Monitoring and Controlling Remote Sensor Networks. IJDSN 5(3): 262-282 (2009)
8. Ioannis Chatzigiannakis, Elisavet Konstantinou, Vasiliki Liagkou, Paul G. Spirakis: Design, Analysis and Performance Evaluation of Group Key Establishment in Wireless Sensor Networks. Electr. Notes Theor. Comput. Sci. 171(1): 17-31 (2007)
9. Ioannis Chatzigiannakis, Andreas Strikos: A decentralized intrusion detection system for increasing security of wireless sensor networks. ETFA 2007: 1408-1411
10. Ioannis Chatzigiannakis, Georgios Mylonas, Sotiris E. Nikolettseas: jWebDust : A Java-Based Generic Application Environment for Wireless Sensor Networks. DCOSS 2005: 376-386
11. Virtualizing testbeds to support large-scale reconfigurable experimental facilities T Baumgartner, I Chatzigiannakis, M Danckwardt, C Koninis, A Kröller, European Conference on Wireless Sensor Networks, 210-223, 2010
12. A web services-oriented architecture for integrating small programmable objects in the web of things O Akrivopoulos, I Chatzigiannakis, C Koninis, E Theodoridis 2010 Developments in E-systems Engineering, 70-75, 2010
13. Open source IoT meter devices for smart and energy-efficient school buildings L Pocero, D Amaxilatis, G Mylonas, I Chatzigiannakis HardwareX 1, 54-67, 2017
14. Elliptic curve based zero knowledge proofs and their applicability on resource constrained devices I Chatzigiannakis, A Pyrgelis, PG Spirakis, YC Stamatiou 2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor ..., 2011
15. On the deployment of healthcare applications over fog computing infrastructure O Akrivopoulos, I Chatzigiannakis, C Tselios, A Antoniou 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC ..., 2017
16. Brown, Eric (13 September 2016). "Who Needs the Internet of Things? . Linux.com
17. Brown, Eric (20 September 2016). 21 Open Source Projects for IoT . Linux.com
18. Internet of Things Global Standards Initiative . ITU.
19. Hendricks, Drew. The Trouble with the Internet of Things . London Datastore. Greater London Authority.
20. Wigmore, I. (June 2014). Internet of Things (IoT) . TechTarget.
21. Developments in the low power, low cost wireless transmitting devices are promising in the area of IoT due to its long battery life and efficiency.
22. Eric Brow Who need internet of Things? (2016)
23. Raji, Reza S. Smart networks for control. IEEE spectrum 31.6 (1994): 49-55
24. Ashton, K. (22 June 2009). That 'Internet of Things' Thing . Retrieved 9 May 2017



25. Gubbi, Jayavardhana, et al. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems* 29.7 (2013): 1645-1660.
26. Siano, Pierluigi. "Demand response and smart grids—A survey." *Renewable and sustainable energy reviews* 30 (2014): 461-478.
27. David B Gray Tesla switches on giant battery to shore up Australia's Grid
28. Sadeghi, Ahmad-Reza, Christian Wachsmann, and Michael Waidner. "Security and privacy challenges in industrial internet of things." 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC). IEEE, 2015.
29. RF Page, Applications of industrial Internet of things (2019)
30. Kushalnagar, Nandakishore, Gabriel Montenegro, and Christian Schumacher. IPv6 over low-power wireless personal area networks (6LoWPANs): overview, assumptions, problem statement, and goals. No. RFC 4919. 2007.
31. Perera, Charith, et al. "A survey on internet of things from industrial market perspective." *IEEE Access* 2 (2014): 1660-1679.
32. Boyes, Hugh, et al. "The industrial internet of things (IIoT): An analysis framework." *Computers in Industry* 101 (2018): 1-12.
33. N. Kushalnagar, G. Montenegro, C. Schumacher IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals (2007)
34. T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP. Vasseur, and R. Alexander. IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL), 2012.
35. Mario Hermann , Tobias Pentek , Boris Otto, Design Principles for Industrie 4.0 Scenarios (2016)
36. Alma Oracevic, Selma Dilek, suat Özdemir Security in Internet of Things: A Survey (2017)
37. Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." *Journal of machine learning research* 12.Oct (2011): 2825-2830.
38. Firdaouss Doukkali Clustering K-Means Algorithm (2017)
39. Conta, Alex, Stephen Deering, and Mukesh Gupta. Internet control message protocol (icmpv6) for the internet protocol version 6 (ipv6) specification. No. RFC 4443. 2006.
40. Shakeel, P. Mohamed, et al. "Maintaining security and privacy in health care system using learning based deep-Q-networks." *Journal of medical systems* 42.10 (2018): 186.
41. "The Enterprise Internet of Things Market". Business Insider. 25 February 2015. Retrieved 26 June 2015.
42. Gilchrist, Alasdair. *Industry 4.0: the industrial internet of things*. Apress, 2016.
43. Lee, Jay, Behrad Bagheri, and Hung-An Kao. "A cyber-physical systems architecture for industry 4.0-based manufacturing systems." *Manufacturing letters* 3 (2015): 18-23.
44. Weber, Rolf H. "Internet of Things—New security and privacy challenges." *Computer law & security review* 26.1 (2010): 23-30.
45. Jing, Qi, et al. "Security of the Internet of Things: perspectives and challenges." *Wireless Networks* 20.8 (2014): 2481-2501.
46. Michie, Donald, David J. Spiegelhalter, and C. C. Taylor. "Machine learning." *Neural and Statistical Classification* 13 (1994).
47. Cover, Thomas M., and Peter E. Hart. "Nearest neighbor pattern classification." *IEEE transactions on information theory* 13.1 (1967): 21-27.
48. Kanungo, Tapas, et al. "An efficient k-means clustering algorithm: Analysis and implementation." *IEEE Transactions on Pattern Analysis & Machine Intelligence* 7 (2002): 881-892.

*to my family, for making this journey possible,  
to all my friends for supporting me,  
to BEST for making it wonderful.*

**“It always seems impossible until it’s done.”**

– Nelson Mandela