Security in Smart Cities:
Adapting Castalia to Simulate Attacks on Deployed Heterogeneous WSNs

Jaume Guasch, *MISTIC student, UOC,*

Abstract

The convergence of the Smart Cities and the Internet of Things open a broad set of opportunities in the provision of new services and the development of new products, leading to better managed cities and more informed citizens that take profit from the information ubiquity.

However, some concerns arise from privacy and security that might condition the acceptance of those technologies. Research efforts have to be put in place to ensure the proper levels of confidence at feasible costs. The security of wireless communications has become a crucial success factor in the deployment of sensor networks by means of more robust access protocols and reliable anomaly detection algorithms.

This work focuses on the necessary provisions taken to simulate existing node networks already deployed as part of the Barcelona's Smart City structure, using real data to construct messages and adapting results from different scenarios to be used later in anomaly detection architectures.

Index Terms

Smart Cities, WSN, Networks Simulation, Castalia Simulator, Information Security, Attacks Detection

1 Introduction

Human population tends to live in growing cities and metropolis. According to the UN, in 2007 the urban population exceeded the rural population and it is expected that this tendency will continue. It is anticipated that by 2030, 60% of world population will already live in large cities[1].

The use and extension of the Information and Communication Technologies (ICT) opens a number of opportunities in the field of cities management. During the 90's decade, the concept of 'Smart City' arose as a set of initiatives that leveraged the intensive use of ICT to bring improvement elements to the citizens, to the resources and to the city management. They become an open field for Innovation [1], providing 'intelligence' to the economy, to people, to the governance, to mobility, to the environment and to life dimensions [2]. On the other hand, the rapid evolution of cities and devices might pose several issues related to information security and users privacy [3]. Additionally, the increasing interconnection of systems even though results in a greater coordination between organizations, in the optimization of resources, in a better management and in a reduction in costs, also raises some challenges related to security of infrastructures that might maintain a physical dependence which may be critical [4].

Amongst the most studied issues, privacy and security are highlighted in the literature. While the first one describes strategies that seek to ensure the different dimensions of the user privacy in front of the services offered by the Smart City [5], in the second, the researchers focus on

[1] World urbanization prospects. the 2005 revision, pop. division, Department of economic and social affairs, UN

managing heterogeneous information sources in the ’smart’ environment and the confluence of a growing and diverse number of data generation elements [6].

In that sense, one key element in the ’smart’ environment is the sensor network as the collection of diverse sensors gathering physical information and sending it to upper layers by means of certain topology arrangement. Sensor networks tend to be heterogeneous because, apart from being designed and installed by different suppliers, they respond to different aims and purposes. In addition, data and control information have several levels of access depending on who physically administers the devices and who exploits the collected information. Thus, the analysis of the data associated with various sensor networks opens a field of study in order to understand their type, possible menaces and, consequently, the impact on security.

2 BACKGROUND AND RELATED WORK

2.1 Smart Cities and IoT

The literature review has implied a deep immersion in the field of Smart Cities. Smart Cities represent the confluence between the intelligent services that big cities or metropolis need and the growing use of ICT as seen in [1]. This is a very broad and prolific field which brings together different research disciplines, covering aspects as diverse as the supplied services, the study of real experiences, in depth study and development of specific technologies and the study of the main concerns that might pose resistances to the deployment and acceptance by users.

Large part of the consulted literature is primarily descriptive. Many authors publish papers where a detailed analysis about the provided services is conducted, discussing the relationship between citizens and administration, presenting the available technologies and justifying the standardization requested to allow systems interoperability. Layered models are presented introducing the concept of security while analysing the aspects such as the impact in infrastructure and the privacy preservation in the service provision [7].

A significant part of de literature describes actual implementation experiences from cities where intelligent services and networks have been deployed. A complete description of the chosen structure, the sensor network, the operating systems and the data models is conducted while pointing out new services and future trends. One example is Barcelona [8], considered one of the leading smart cities all over Europe and an interesting approach combining smart districts, living labs, e-Services, Open Data and providing examples of district transformation as the 22@ successful implementation. Another example is Santander [9] with SmartSantander, one of the largest smart city test-beds in Europe and a clear example of platform integration towards the effective use of the large amounts of data collected from all the deployed sensors.

The bibliography also points to another research field that is converging rapidly with smart cities. From the new trends in the so-called Future Internet [10], the Internet of Things (IoT), [11] is the result of a combination of elements such as ubiquitous computing, sensors technology, wireless communications, Internet and the embedded devices. IoT presents common factors with smart cities and the ability to bring different elements that are connected to each other, to other networks or the Internet from anywhere, including users provided, among others, with smart phones [12].

2.2 Wireless Sensors Networks

A common area between the IoT and the Smart Cities is Sensor Networks. Sensor networks are usually of diverse types and from different manufacturers and might present doubts related to ensuring security, among others issues [3]. Security is contemplated in the broadest sense, covering availability, the integrity and capacity of authorized access. The requirements of both low cost and low power consumption of sensor networks that characterize the IoT networks pose an additional challenge in adapting strategies to ensure security by means of, for instance, the use of specific communication protocols and new developments of lightweight cryptographic algorithms and the design of specialized hardware [13].

Current developments in both smart cities and in IoT are nourished primarily from an extensive bibliography in the field of sensor networks [14] and particularly from Wireless Sensor Networks (WSN). Security in IoT and in Smart Cities is directly related to security in WSNs. The WSNs, unlike wired networks, suffer from additional limitations that make it challenging to ensure their security when facing attacks and / or incidents in their function and operation [15]. This aspect becomes notoriously relevant due to the increasing number of applications where security is critical and might condition policy-makers' acceptance due to a lack of confidence in the system reliability and data integrity, or generate final users resistances related to privacy issues.

As mentioned before, both the networks and the sensor nodes are heterogeneous and are often controlled directly by the service provider or by the company that deployed the network. These providers will look for economic solutions that meet the specifications described in the tenders, delegating the network end user the responsibility of detecting irregularities in its operation caused by incidences or attacks. Ensuring network security and data integrity is a fundamental requirement and will require the development of mechanisms to infer security problems or data reliability issues due to any anomalous operation of the facility, detecting the presence of a possible attack [16,17,18,19] and even being able to estimate the level of security of a given WSN [20].

These studies are not well suited to be performed directly in the field because they might affect associated services and would be limited by the restrictions commonly found in real conditions and related to ambient electrical noise, radio interferences, physical access limitations and radio-frequency emission regulations amongst others. Conversely, the ability to simulate real networks in a lab environment can be a good compromise between the fidelity of the model and the ability to emulate a wide range of situations and scenarios.

3 Pursued Objective

Information about several groups of nodes from different brands, characterized by a list of individual sensor node references and their GPS positions has been obtained for the city of Barcelona. The actual received data from the sensors during an extensive period up to 14 days has also been supplied. Unfortunately, no additional information about network routing or the presence of additional routing nodes or gateways has been indicated.

The main objective of the present work has been, as a design and creation project, to set-up a simulation environment in order to emulate a real network of already existing nodes. The simulation has been based on the available information regarding node position and the existing characteristics of the nodes and sensors. These networks, formed out from a diverse number of nodes, are from various manufacturers and respond to different physical parameter measurements.

The network has been configured and the nodes have been programmed to send the corresponding messages to the network. The available received messages (once sent from the actual corresponding nodes) are used to be sent at the same rate and intervals they were originally received. This approach, instead of programming the nodes to send messages following certain statistical distribution, tries to be a more realistic implementation, on the simulator, of the actual network characterized by a range of different data volumes and cadences.

Different scenarios to emulate anomalies and attacks might be simulated in order to compare the effects on the received data when applied. The information coming from every simulation has to be processed accordingly in order to feed a detection system that would learn from every scenario in different time intervals and infer whether an attack occurred. The detection system and the work associated with it in that area is out the scope of this work.

4 DEVELOPMENT

4.1 Simulation Environment

The simulation environment is build using Castalia 3.2 [21] over OMNeT++ 4.6 [22]. OMNeT++ is an object-oriented modular discrete event network simulation framework with a generic architecture suitable for being used in different domains and applications. OMNeT++ provides the tools required to write any kind of event driven simulation.

The basic element of OMNeT++ is the module. Modules may be simple or compound. Simple modules are the lowest level of hierarchy and its behaviour is programmed in C++. Compound modules are formed by a number of other simple and / or compound modules. The number of hierarchy levels is not limited and modules might be re-used into other modules.

Modules are connected to other modules via gates that send and receive messages that are formed by arbitrary data structures. Modules can pass messages through gates or send them directly to the destination module. The latter, facilitates the construction of wireless networks. A simulation model is composed by different simple and compound modules, that are interconnected in order to pass messages among them. The structure is defined in an specific language called NED and configured in files with the same extension name. Simulation parameters are contained into an initialization file named `omnetpp.ini`.

Castalia is a simulator designed for Wireless Sensor Network (WSN) and Body Area Networks (BAN) that are both characterized by low power embedded devices. Castalia is constructed over the OMNeT++ platform and pre-defines several simple and compound modules that allow the researcher to easily simulate WSNs and BANs with different structures and parameters. The user might also modify existing modules or implement new ones to include new protocols or add new applications with specific functionality not covered by default in Castalia. Figure 1 shows the basic modules defined in Castalia:

- **Physical Process**: These modules represent the physical magnitudes that are measured and controlled by the nodes. Nodes send message to obtain readings. Any node might have different physical processes representing diverse measure magnitudes.

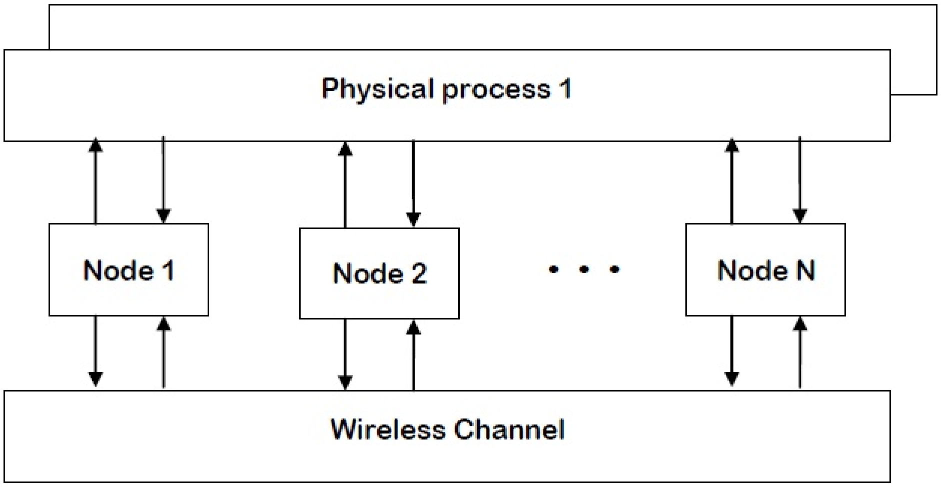

Fig. 1: Castalia Basic Modules (source: Castalia user manual)

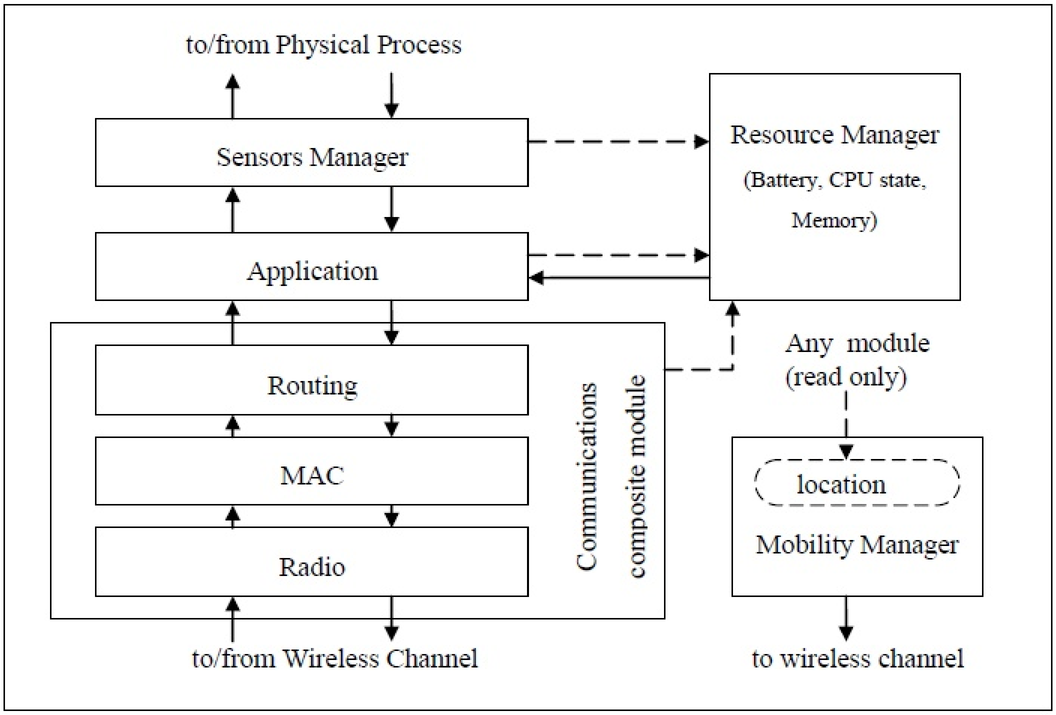

Fig. 2: Castalia Node Module (source: Castalia user manual)

- **Wireless Channel**: Emulates the wireless medium that connects nodes. The model simulates a variety of conditions regarding attenuation, fading, temporal variations and signal loss.
- **Nodes**: Compound modules formed by other modules that implement the functionality of a communications node. Figure 2 shows the node structure and the constitutive modules currently build.in. Nodes are connected to other nodes by means of the Wireless Channel module and get physical information from the Physical Process modules. Nodes are composed by the following modules:
  - **Sensor Manager**: Simple module responsible for the physical access functionality

- **Application Module**: Simple module responsible of implementing the node application level connecting the Sensor Manager with the Communications module.
- **Communications Module**: This is a compound module responsible for the implementation of the functionality related to the Routing, the Media Access Control (MAC) and the Radio behaviour of the node.
- **Resource Manager**: Manages the energy consumption of the node and allows controlling the node clock drift.
- **Mobility Manager**: This module is responsible for the movement control of nodes in the space if that functionality is required.

The definition of these modules and sub-modules follows the same hierarchy of files that determines their structure. Castalia as OMNeT++ allows the user to define new modules and new capabilities using C++ language for simple module behaviour description and NED language for modules structure definition.

4.2 Baseline Data

As previously commented, the supplied information is constituted of a series of files containing firstly the name and position for the nodes of different brands and secondly, the nodes data received during a given period of time of approximately 14 days. The names of nodes and their position are defined in a csv file with the structure shown bellow (actual brand names omitted):

```
x,y,Node_ID,Provider
2.1303350072,41.3852070789,MIC0001,BRAND_1
2.1300723767,41.3846812626,MIC0002,BRAND_1
(...)
2.1306075408,41.3851881008,T240714,BRAND_2
2.1299546229,41.3842444442,T240711,BRAND_2
(...)
2.1305602148,41.3840314145,71330,BRAND_3
2.1303745153,41.3838977008,71315,BRAND_3
(...)
```

On that file, `x` and `y` are the GPS coordinates for each node, `Node_ID` are the node identifiers and `Provider` indicates the node brand name. Files (one per brand) containing the received event messages follow the structure (see two examples for 2 different brands):

```
nodeId,"message_app",time,"timestamp_message"
MIC0003,"64.9",1442698020,"2015-09-19 23:27:00.000"
MIC0010,"65.2",1442697960,"2015-09-19 23:26:00.000"
MIC0007,"65.7",1442697960,"2015-09-19 23:26:00.000"
(...)

nodeId,"message_app",time,"timestamp_message"
"TA120-T240711-N","65.1","1442699927.340000","2015-09-19 23:58:47.340"
"TA120-T240708-N","68.5","1442699921.387000","2015-09-19 23:58:41.387"
"TA120-T241198-B",100,"1442699913.466000","2015-09-19 23:58:33.466"
(...)
```

The field `nodeId` corresponds to the node identifier, the field `message_app` contains the received value, the field `time` contains the instant time when the message was sent and is coded using the Linux epoch time[2]. Finally, `timestamp_message` represents previous `time` value in a date and time format.

Additionally, kml files used to represent spatial position of nodes in Google Earth are also available for all brands. Those files visually helped defining the routing among nodes as will be seen later but are not explicitly used along the simulations.

4.3 Simulation Set-up

In order to handle the given information and prepare the required files to conduct the simulations, a series of Python scripts are developed:

```
$ python3 ordena.py <file_name>.csv
```

The csv files that contain the received event messages (for all nodes of each brand) had to be ordered in increasing time codes. This arrangement would be more convenient for the simulation later. Timecodes for different brands are all set to seconds format and values for `message_app` are also normalized according to table 1 to simplify the message construction later in the Castalia application as seen in the next chapter. Most of the sensors send numerical values while some send logical values or JSON[3] messages that are converted to numbers for a more straightforward management when constructing node messages.

Original Event Value	**Converted Value**
True	0.1
False	0
″{″(...)	999.9

TABLE 1: message_app conversion values

The second script takes the nodes definition file, the number of intervals and the interval duration the simulation will have as input parameters:

```
$ python3 setup_simulation.py <nodes_file>.csv #intervals duration
```

The simulation configuration file `omnetpp.ini` and the data files for the nodes are created by the script. The purpose and use of these files is explained in the next section. As node position is given in GPS coordinates and the simulator requires the node placement with X,Y,Z coordinates in meters (Z is considered zero due to the absence of height data for the nodes), a conversion is obtained using the following approximation[4]:

```
  X[m] = 111.195*(longitude-longitude_origin)[deg]*COS(latitude[rad])
  Y[m] = 111.195*(latitude-latitude_origin)[deg]
```

Taking for instance the 10 nodes from brand BRAND_1, defined in the initial csv file, we can see the placement in Google Earth (Figure 3) and in a X,Y coordinates system (Figure 4).

[2] Either coded in s or μs. See https://en.wikipedia.org/wiki/Unix_time

[3] For additional information on JavaScript Object Notation, see: http://www.json.org/

[4] Distance between two earth parallels might be approximated by: ¼$(2\pi r)/90$ where r is the earth radius. X is directly obtained and Y is corrected by the cos(latitude). Resulting errors were inferior to 1m when compared to the results using: http://www.whoi.edu/marine/ndsf/cgi-bin/NDSFutility.cgi?form=0&from=LatLon&to=XY

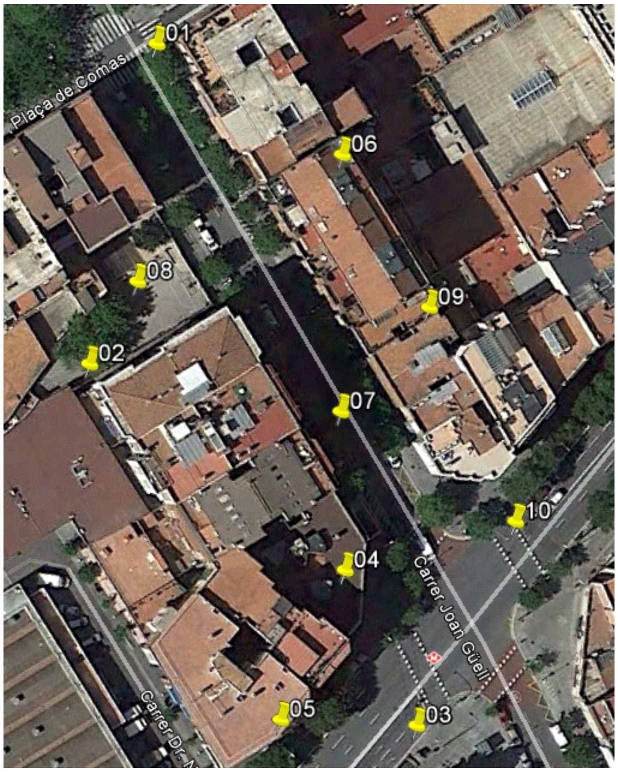

Fig. 3: BRAND_1 nodes placement on a map (.kml file on Google Earth)

The resulting `omnetpp.ini` file contains the required configurations to run the simulation for all the selected nodes. A node 0 is defined as receiver or 'sink' and it is positioned in the centre of the area (see fig. 4) determined by all the nodes. An attacker node is also defined with index [number of nodes + 1] that will have to be placed manually by the user.

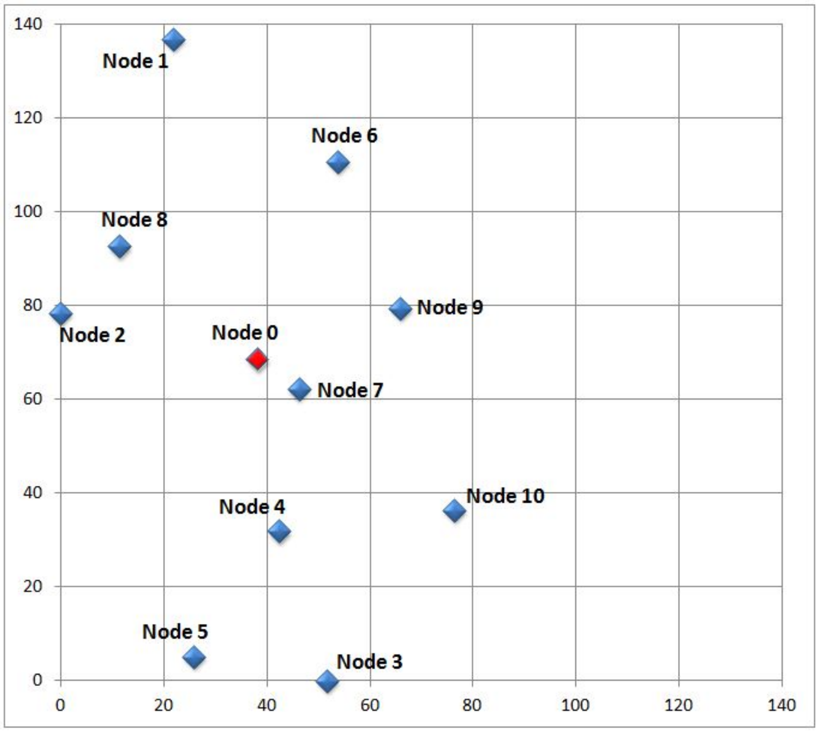

Fig. 4: BRAND_1 nodes and node 0 placement on an X,Y system (axes in m)

4.4 Developments in Castalia

As commented earlier, one initial requirement was to send the available data values, that were received from the actually deployed nodes, by the simulated nodes, but no convenient functionality is implemented in Castalia yet, as described bellow. This has been one of the major challenges to face during the simulation set-up.

One first approach to achieve the objective consisted on sending the values directly from the Castalia's Physical Process module, a level at which values can be assigned at specific nodes. Castalia's `CustomizablePhysicalProcess` allows the assignment of constant values to the nodes `(DIRECT_NODE_VALUES):`

```
SN.physicalProcessName = "CustomizablePhysicalProcess"
SN,physicalProcess.inputType = 0
SN.physicalProcess[*].directNodeValueAssignment
SN.physicalProcess.directNodeValueAssignment = "0:0 1:25 2:30 3:36
49:11 50:10 51:16 52:13 53:20 54:18 55:20 56:25 57:27 58:26 59:24
60:24 61:25 62:25 ..."
```

However, this approach would not fulfil entirely the goal because values are constant per node while the aim was to send different values over time. A second approach consisted in assigning node number, position and value to the Physical Process `(SCENARIO_BASED):`

```
string source_0 = default("0 10 10 30.5; 5 10 10 45; 12 10 10 7.3;
5 10 10 30; ...");
```

This method implies limitations regarding the number of values that a node could send based on the maximum length that the string `source_0` would allow. Some node files contained more than 6.000 events, hence this method did not seem to be very convenient either.

Finally, the third method listed in Castalia for value assignment `(TRACE_FILE)` that seemed to be provided to read an external file with physical values is not yet implemented in the simulator code, despite numerous requests on Castalia user forums.

Since the main interest was to send values from the available files from the nodes, the decision of modifying the application module was taken for simplicity. The application level can read the time and data value to be sent and create the packet. On that basis, a new application module called `ThroughputTestNEW` was derived from the default `ThroughputTest` writing new .cc, .h and .ned files accordingly.

Therefore, the new application includes a build-in function that reads the values to be sent for every node from the corresponding external file containing time to next event and value to be sent at that instant:

```
int ThroughputTestNEW::readInputFile
(string file, map<long,int>& temps, map<long,int>& valors)
{
     std::ifstream filestream;
     std::string line;
     filestream.open(file, std::ifstream::in);
     int i = 0;
     string subline1, subline2;
     if(!filestream) {
         throw cRuntimeError("Error opening the node parameters file");
         trace()<<"Error opening the sensor parameters file: "<<file;
         return -1;
     } else {
         while ( getline(filestream, line) ) {
             if (line.find(’#’) == 0) {
                 continue;   // ignore comment lines
             }
             else {
                 subline1 = line.substr(0,line.find(’,’));
                 subline2 = line.substr(line.find(’,’)+1,line.length()-1);
                 stringstream(subline1) >> temps[i];
                 stringstream(subline2) >> valors[i];
                i++;
             }
         }
     }
     return i;
}
```

The file is read and values are loaded into `<map>` containers. The function returns the number of overall available events per node. If this value is zero the node has no messages to be sent and will be idle. A timer is added into the application that is fired when the message has to be sent. When that moment arrives, according to the times read from the external file, the application forms the packet and sends it to the Communications Module:

```
case REQUEST_SAMPLE:{
    sensorRead = sensorValues[numberTimesSensed];
    toNetworkLayer(createGenericDataPacket(sensorRead,
    numberTimesSensed), recipientAddress.c_str());
    numberTimesSensed++;
    packetsSent[recipientId]++;
    if (numberTimesSensed+1 > numEvents){
        trace() << "No more samples...";
        cancelTimer(REQUEST_SAMPLE);
    }
    else {
        setTimer(REQUEST_SAMPLE,
        sensorTimes[numberTimesSensed]);
    break;
    }
}
```

If the number of available samples for a given node is achieved, the corresponding timer is cancelled and the node passes to idle state from that moment on. This allows the simulation of heterogeneous nodes with diverse data amounts and cadences.

The attacker node is defined as the higher Id node. The original application timer is slightly modified to serve as the attacker timer according to the node `packet_rate` parameter:

```
case SEND_PACKET:{
    //trace()<<"Sending packet #"<<dataSN;
    toNetworkLayer(createGenericDataPacket(0,dataSN),
    BROADCAST_NETWORK_ADDRESS);
    packetsSent[self]++;
    dataSN++;
    setTimer(SEND_PACKET,packet_spacing);
    break;
}
```

Attacker node sends data packets with value 0 to broadcast and follows the chosen attack definition contained into `omnetpp.ini` configuration file (Several attack definitions might be configured. See next chapter for an execution example of a Jamming Attack):

```
[Config JammingAttack]
SN.node[11].Communication.Radio.TxOutputPower = "0dBm"
SN.node[11].Application.constantDataPayload = 277
SN.node[11].Communication.MACProtocolName = "TunableMAC"
(...)
SN.node[11].xCoor = 50
SN.node[11].yCoor = 80
[Config ContinuousChannelAccessAttack]
SN.node[11].Communication.Radio.TxOutputPower = "-15dBm"
SN.node[11].Application.packet_rate = 246
(...)
```

Finally, in order to run the simulations with the specified intervals of the indicated duration, the `Application.startupDelay` parameter is used together with the `sim-time-limit` to define the requested intervals into the `omnetpp.ini` file:

```
[Config Interval0]
SN.node[*].Application.startupDelay = 0
sim-time-limit = 3600s
[Config Interval1]
SN.node[*].Application.startupDelay = 3600
sim-time-limit = 7200s
[Config Interval2]
SN.node[*].Application.startupDelay = 7200
sim-time-limit = 10800s
```

Previous example configures three intervals of one hour each. Hence, if a node has an event to be sent at absolute time t=3650s, the corresponding message will be sent during `Interval1` execution. Different node networks might require different interval durations as their packets rate might be very different as well (i.e. temperature measure nodes vs. parking lot state nodes) and the convenient election when combined might be challenging.

5 Analysis and Results

5.1 Initial Results and Improvements

In order to test the simulator with the actual data, a first scenario for an initial simulation based on the 10 nodes shown at figures 3 and 4 is arranged. Observe that the `omnetpp.ini` file will define

12 nodes, that is: the sink node or node 0, 10 nodes from the selected brand (BRAND_1) and an attacker node that might be active or not depending on the selected scenario. A configuration of one interval of four hour duration is chosen as example. All nodes send messages to node 0.

```
$ python3 setup_simulation sensors_BRAND_1.csv 1 4
```

The set up script will place node 0 at the centre of the area defined by the selected nodes as shown in figure 4. User might place attacker node where is more convenient for the simulation directly in the `omnetpp.ini` configuration file.

```
x,y,Node_ID,Provider,nextRecipient
2.1303350072,41.3852070789,MIC0001,BRAND_1
2.1300723767,41.3846812626,MIC0002,BRAND_1
(...)
2.1308645241,41.3846908052,MIC0009,BRAND_1
2.1309893575,41.3843021502,MIC0010,BRAND_1
```

The 10 files with the data to be sent by the corresponding nodes will also be created and named `BRAND_1_Node_x.txt`, where `x` is the node number. As an example, for node 1:

```
# Automatically created file for node MIC0001
180,541  # first digit is time to next event, second value to send
3660,662
660,628
(...)
# Events: 6857
# Total Time: 1209300(s)
# Initial Simulation Time at: 1441488480, 2015-09-05 23:28:00.000
```

The configuration file will define node positions, input files and intervals automatically and will get the definition for MAC and attacks from `macs.ini` and `attacks.ini` respectively. For this example simulation a `NoAttack` configuration is used.

```
SN.numNodes = 12

# XY node coordinates [meters]
SN.node[1].xCoor = 21.91
SN.node[1].yCoor = 136.86
SN.node[2].xCoor = 0.00
SN.node[2].yCoor = 78.39
SN.node[3].xCoor = 51.85
SN.node[3].yCoor = 0.00
(...)
SN.node[10].xCoor = 76.50
SN.node[10].yCoor = 36.23

# Sink Node placed at middle of nodes field
SN.node[0].xCoor = 38.25
SN.node[0].yCoor = 68.43

# Files for node values
SN.node[1].Application.fileName = "BRAND_1_Node_1.txt"
SN.node[2].Application.fileName = "BRAND_1_Node_2.txt"
SN.node[3].Application.fileName = "BRAND_1_Node_3.txt"
(...)
SN.node[10].Application.fileName = "BRAND_1_Node_10.txt"

[Config Interval0]
SN.node[*].Application.startupDelay = 0
sim-time-limit = 14400s

[Config NoAttack]
SN.node[11].Application.packet_rate = 0
SN.node[11].Communication.Radio.TxOutputPower = "-15dBm"
SN.node[11].Communication.MACProtocolName = "TunableMAC"
```

All application nodes will use TMAC as MAC protocol except the attacker node, if active, that is assumed not to be part of the same network and will not be related to it.

```
[Config TMAC]
SN.node[0..10].Communication.MACProtocolName = "TMAC"
SN.node[0..10].Communication.MAC.phyDataRate = 250
SN.node[0..10].Communication.MAC.maxTxRetries = 5
SN.node[0..10].Communication.MAC.waitTimeout = 5
SN.node[0..10].Communication.MAC.collisionResolution = 0
```

When the simulation is executed, an output file is created that is converted by Castalia's build-in function `CastaliaResults` into a more readable csv file.

```
$ Castalia -c Interval0,TMAC,NoAttack -o results.txt
$ CastaliaResults -i results.txt -s "^" -n -o 2 > results.csv
```

The resulting output file contains the values defined and collected by the modules behaviour descriptions using the Castalia's methods:

```
declareOutput(appropriate name for the output);
collectOutput(output name, index, label, amountIncreased);
```

Taking, for instance, the number of packets received and the reception rate per node, table 2 summarizes the simulation results for the previously defined interval showing the received packets at node 0 coming from application nodes.

	node 2	node 4	node 7	node 8
#received	4	31	56	43
Rec. rate	0.06	1	0.97	0.94

TABLE 2: Packets received at node 0. First scenario

As seen, reception at node 0 is limited to messages from nodes 2, 4, 7 and 8. Reviewing the map (fig. 3 & fig. 4), we might observe these nodes are closer to node 0 than the others. Power transmission constraints limit the effective distance nodes can communicate directly. In the map some nodes are relatively far from node 0 and packets are lost due to insufficient reception level (i.e. node 1 is more than 60m away from node 0).

A second scenario is designed to improve the reception rate of nodes, implementing a certain routing among them based on their physical placement. Using this strategy, every node sends messages to their neighbour node in a multi hop configuration until message reaches node 0, or sink. More messages are expected to be received counteracting power limitations.

Routing information is eventually added to ease the process as an additional column into the initial csv file to indicate the next recipient for every node. Final recipient will be node 0.

```
x,y,Node_ID,Provider,nextRecipient
2.1303350072,41.3852070789,MIC0001,BRAND_1,6
(...)
2.1308645241,41.3846908052,MIC0009,BRAND_1,7
2.1309893575,41.3843021502,MIC0010,BRAND_1,4
```

The script `Setup_simulation.py` is modified accordingly to compile this information to `omnetpp.ini` file creating two different configurations:

```
[Config NoMultihop]
SN.node[*].Application.nextRecipient = "0"

[Config Multihop]
# Routing for BRAND_1 nodes
SN.node[1].Application.nextRecipient = "6"
SN.node[2].Application.nextRecipient = "8"
(...)
SN.node[10].Application.nextRecipient = "4"
```

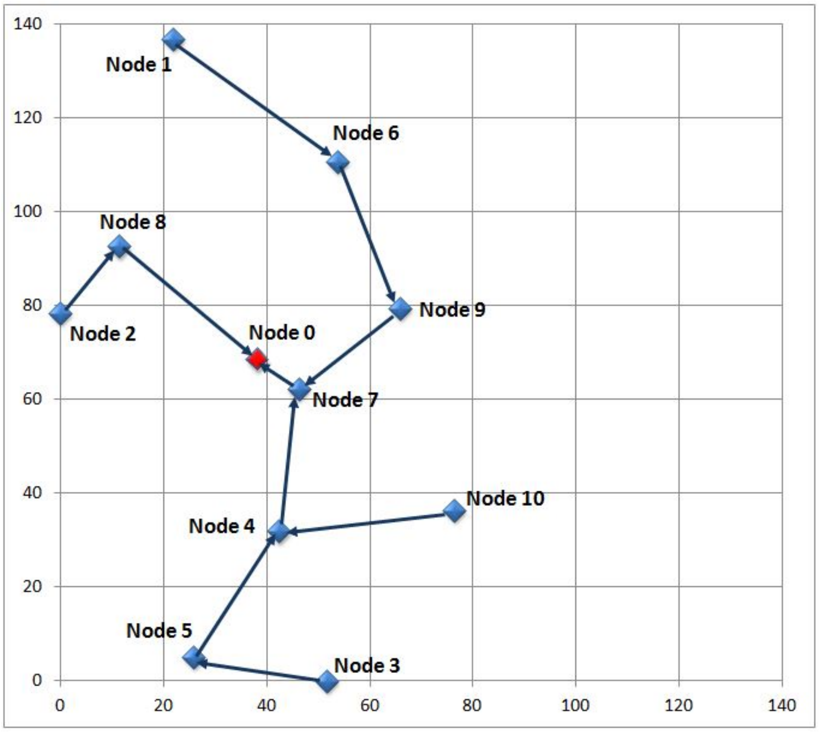

Fig. 5: BRAND_1 nodes routing to node 0

The programmed routing is graphically shown at figure 5. A new field has to be added to retain the origin of every packet when created. The ApplicationPacket.msg file is modified to accommodate the new variable into the packet structure:

```
packet ApplicationPacket {
    AppNetInfoExchange_type appNetInfoExchange;
    string applicationID;
    unsigned int sequenceNumber;
    double data;
    string originalSourceId; //added field to register origin node
}
```

The application ThroughputTestNEW initializes this new variable in order to keep track of the node origin every time a packet is created:

```
newPacket->setOriginalSourceId(SELF_NETWORK_ADDRESS);
toNetworkLayer(newPacket, recipientAddress.c_str());
```

When packets arrive to node 0, the origin node is registered because we want to keep track of the reception ratios per node whether or not the simulation uses a multi hop routing:

```
collectOutput("Packets received per node", originalSourceId);
packetsReceived[originalSourceId]++;
```

Repeating the previous simulation settings using the routing definition and the multi hop configuration a clear improvement is achieved, although reception rate is not 100%. (See reception results in table 3):

```
$ Castalia -c Interval0,TMAC,NoAttack,MultiHop -o results.txt
$ CastaliaResults -i results.txt -s "^" -n -o 2 > results.csv
```

	node 1	node 2	node 3	node 4	node 5	node 6	node 7	node 8
#received	19	62	44	30	79	16	57	45
Rec. rate	0.45	0.97	0.62	0.97	0.9	0.57	0.98	0.98

TABLE 3: Packets received at node 0. Multi hop scenario

It is to be noted that node 9 has no packets to send since MIC0009 node had no messages into the originally supplied events file. More interestingly, node 10 does not achieve to send packets to node 0 in any situation due to radio congestion at node 4. The simulation trace file shows node 10 sending a packet to node 4 to be forwarded to node 7 and almost at the same instant node 5 is sending a packet to node 4 too:

```
2100.046945048995SN.node[10].Application
Data Packet sent #0 from node: 10 with value: 599 to node: 4
2100.074763843382SN.node[5].Application
Data Packet sent #18 from node: 5 with value: 675 to node: 4
2100.34433361849SN.node[4].Application
Forward packet from node 5 & origin: 5 to node 7 with value 675
```

Node 4 does not receive the packet from node 10 and only forwards the packet received from node 5 to next recipient node. When node 10 sends its packet to node 4, the radio of node 4 is not ready as seen in the trace file of the radio module for node 4:

```
2100.5675218264 SN.node[4].Communication.Radio START signal from node 10,
received power -93.6348dBm
2100.5675218264 SN.node[4].Communication.Radio Failed packet
(WC_SIGNAL_START) from node 10, radio not in RX state
2100.5681298264 SN.node[4].Communication.Radio END signal from node 10
```

Node 10 will retry sending the packet to node 4 according to the defined maximum number of retries and will finally discard sending that packet:

```
SN.node[0..10].Communication.MAC.maxTxRetries = 5
```

Similar situation is reflected by nodes 1, 3 and 6 with reception rates bellow 90% that would be expected for a set of nodes using a MAC protocol as TMAC.

Congestion in certain nodes has been a big issue not totally solved in order to completely emulate the original network with the supplied event files. Adding intermediate nodes has finally been discarded mainly because no information is available about the existence of any additional node in the actual network and because simulations showed that congestion problem prevails or even increases with intermediate nodes. Further work might be undertaken in that direction in order to add more intelligence to the application layer, despite the assumed node capacity limitations, and eventually improving the selected MAC protocol.

Previous simulation is now repeated applying an attack that is designed to affect the transmission medium. The attacker node uses a different protocol than the rest of nodes because it is supposed not related to the existing network and free to transmit independently of the channel state. This transmission medium attack might be assimilated to a Jamming Attack.

```
[Config JammingAttack]
SN.node[11].Communication.Radio.TxOutputPower = "0dBm"
SN.node[11].Application.constantDataPayload = 277
SN.node[11].Communication.Routing.maxNetFrameSize = 2500
SN.node[11].Communication.MAC.maxMACFrameSize = 2500
SN.node[11].Communication.Radio.maxPhyFrameSize = 2500
SN.node[11].Communication.MACProtocolName = "TunableMAC"
SN.node[11].Application.packet_rate = 75
SN.node[11].xCoor = 50
SN.node[11].yCoor = 80
```

Attacker node is manually placed at coordinates x=50 and y=80. Table 4 summarizes the effect of the applied attack to the simulation network.

	node 1	node 2	node 3	node 4	node 5	node 6	node 7	node 8
#received	17	59	23	30	24	15	58	46
Rec. rate	0.4	0.92	0.32	0.97	0.27	0.54	1	1

TABLE 4: Packets received at node 0. Multi hop with Jamming Attack

Table 4 results compared with table 3 values show the effects of the applied attack at the reception values mainly for nodes 3 and 5. The number of received packets and the corresponding reception rate are clearly reduced when the attack is applied.

Different additional configurations might be used in order to emulate other attacks in diverse conditions focusing on affecting selected areas of the network, the sink node or selected sub-networks corresponding to an specific brand or functionality.

5.2 Simulation and Output Files

The final aim of the present work has been to supply prepare simulation result files in a convenient format to be used in anomaly detectors. These detectors should be trained with simulation data coming from several intervals and the actual data sent by the nodes in order to emulate the real network formed by different nodes from various brands.

These files will be the result of simulations run in different scenarios. That is, without any attack for certain intervals and applying selected attacks to other intervals to infer whether the system is able to detect the anomalies when they occur and even being able to classifying the type of attack, if any.

As mentioned earlier, this later analysis is out of the scope of this work. However, the required provisions to compile the `CastaliaResults` csv file represented the last development of the present work. The goal has been the creation of a single table for every simulation containing one row per interval and scenario (no attack, attack of type 1, attack of type 2, etc.) and columns gathering information about the following parameters for every node (with one column per node):

- Packets received at node 0
- Reception loss at node 0
- Reception rate at node 0
- Consumed energy during simulation
- Radio Tx packets
- MAC sent packets breakdown attending to ACK, CTS, DATA, RTS & SYNC values
- Radio RX packets breakdown attending to Failed with NO interference, Failed with interference, Failed, below sensitivity, Failed, non RX state, Received despite interference & Received with NO interference values

A Python script has also been created on that purpose:

```
$ python3 convertResultsFile.py #Intervals #Scenarios
```

Where `#Intervals` is the number of intervals the simulation set-up defined (duration is not relevant for the file conversion), and `#Scenarios` is the number of cases that the simulation envisaged (several attack types, multi hop or not, different types of MAC, etc).

All developed applications and scripts required to perform simulations can be found at [23] with instructions on how to install them in Castalia. An example of the final output results table is also available for a 50 intervals and 1 scenario case for a network formed by 55 nodes from 3 different brands.

6 Conclusion and Further Work

This project shows the feasibility of adapting Castalia to simulate actual deployed WSNs taking supplied data from external files to construct and send messages. Even though some congestion prevails even under ideal conditions and after the improvement achieved with the multi hop routing, the effect of applied attacks might be clearly seen from the simulation results and further work will show whether they can be inferred from anomalies detection.

In that sense, last project efforts have been devoted to convert simulation results file into a more convenient format requested to feed anomaly detection systems.

This work might be mainly considered as a design and creation project but findings and results do open new lines of activity that could be undertaken afterwards:

- Use the current developments to conduct systematic simulations around the supplied baseline data to feed selected anomaly detectors with different scenarios to evaluate the detection capacity.
- Revisit congestion issues in depth to fully understand the causes of packets loss and take the necessary provisions to achieve higher reception rates in dense networks.
- Study the simulation randomness in cases where, for instance, adding a passive node affects the simulation results, due to slightly different initialization values for several simulation parameters as channel attenuation or clock drift.
- Define and characterize different attacks and test the effect upon position, MAC and mobility among other factors. Include the possibility of more than one attacker node and define a more systematic way to place attacker nodes on the field.
- Implement Collection Tree Protocol (CTP)[5]. CTP is frequently used in WSNs and it is widely cited and studied in the literature. A Castalia implementation[6] of CTP exists but it is not maintained since 2012. Some development effort should be put in place to update the CTP implementation to the latests protocol revisions and adapt it to run under the latest versions of Castalia and OMNeT++.

7 Acknowledgments

The author would like to thank UOC University and specially KISON research group professors and consultants as well as the MISTIC program students for their help, collaboration and constructive comments on the present work.

References

[1] Schaffers, H., Komninos, N., Pallot, M., Trousse, B., Nilsson, M., & Oliveira, A. (2011). Smart cities and the future internet: towards cooperation frameworks for open innovation. In The future internet (pp. 431-446). Springer Berlin Heidelberg.

[2] Presser M, Hernndez Gmez LA, Pettersson J, HernndezMuoz JM, Bernat Vercher J, Galache JA, et al. Smart Cities at the forefront of the future internet. In The future internet (pp.447-462). Springer Berlin Heidelberg.

[3] Elmaghraby, A.S. & Losavio, M.M. 2014, ”Cyber security challenges in Smart Cities: Safety, security and privacy”, Journal of Advanced Research, vol. 5, no. 4, pp. 491-497. Jul. 2014.

[4] V. Garcia-Font, C. Garrigues, H. Rifà-Pous. Seguridad en Smart Cities e Infraestructuras Críticas. RECSI 2014, Alicante, 2-5 Septiembre 2014.

[5] https://sing.stanford.edu/gnawali/ctp/ (last access: Nov. 30, 2015)
[6] https://code.google.com/p/ctp-castalia/ (last access: Nov. 30, 2015)

[5] Martinez-Balleste, A.; Perez-Martinez, P.A.; Solanas, A., ”The pursuit of citizens’ privacy: a privacy-aware smart city is possible,” Communications Magazine, IEEE , vol.51, no.6, pp.136,141, Jun. 2013.

[6] Mitton, N., Papavassiliou, S., Puliafito, A. & Trivedi, K.S. 2012, ”Combining Cloud and sensors in a smart city environment”, EURASIP Journal on Wireless Communications and Networking, vol. 2012, no. 1, pp. 1-10.

[7] D. Rebollo-Monedero, A. Bartoli, J. Hernández-Serrano, J. Forné, and M. Soriano, ”Reconciling privacy and efficient utility management in smart cities,” Transactions on Emerging Telecommunications Technologies, vol. 25, no. 1, pp. 94-108, Jan. 2014.

[8] T. Bakici, E. Almirall, and J. Wareham, ”A Smart City Initiative: the Case of Barcelona,” Journal of the Knowledge Economy, vol. 4, no. 2, pp. 135-148, Jun. 2013.

[9] Sanchez L, Galache JA, Gutierrez V, Hernandez JM, Bernat J, Gluhak A, et al. SmartSantander: The meeting point between Future Internet research and experimentation and the smart cities. Future Network & Mobile Summit (FutureNetw 2011): Warsaw, Poland, 15 - 17 June 2011. Piscataway, NJ: IEEE, 2011.

[10] J. M. Hernández-Muñoz, J. B. Vercher, L. Muñoz, J. A. Galache, M. Presser, L. A. H. Gómez, and J. Pettersson, Smart cities at the forefront of the future internet. Springer, 2011.

[11] E. Borgia, ”The Internet of Things vision: Key features, applications and open issues,” Computer Communications, vol. 54, pp. 1-31, Dec. 2014.

[12] L. Atzori, A. Iera, and G. Morabito, ”The Internet of Things: A survey,” Computer Networks, vol. 54, no. 15, pp. 2787-2805, Oct. 2010.

[13] S. Cirani, G. Ferrari, and L. Veltri, ”Enforcing Security Mechanisms in the IP-Based Internet of Things: An Algorithmic Overview,” Algorithms, vol. 6, no. 2, pp. 197-226, Apr. 2013.

[14] G. Hancke, B. Silva, and G. Hancke, Jr., ”The Role of Advanced Sensing in Smart Cities,” Sensors, vol. 13, no. 1, pp. 393-425, Dec. 2012.

[15] T. Kavitha and D. Sridharan, ”Security vulnerabilities in wireless sensor networks: A survey,” Journal of information Assurance and Security, vol. 5, no. 1, pp. 31-44, 2010.

[16] Z. Banković, D. Fraga, J. M. Moya, and J. C. Vallejo, ”Detecting Unknown Attacks in Wireless Sensor Networks That Contain Mobile Nodes,” Sensors, vol. 12, no. 12, pp. 10834-10850, Aug. 2012.

[17] W. Xu, W. Trappe, Y. Zhang, and T. Wood, ”The feasibility of launching and detecting jamming attacks in wireless networks,” in Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing, 2005, pp. 46-57.

[18] A Este, F Gringoli, L Salgarelli. ”Support vector machines for TCP traffic classification.” Computer Networks 53.14 (2009): 2476-2490.

[19] Kaplantzis, Sophia, et al. ”Detecting selective forwarding attacks in wireless sensor networks using support vector machines.” Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on. IEEE, 2007.

[20] A. Ramos and R. Filho, ”Sensor Data Security Level Estimation Scheme for Wireless Sensor Networks,” Sensors, vol. 15, no. 1, pp. 2104-2136, Jan. 2015.

[21] Information and documentation at: https://castalia.forge.nicta.com.au/index.php/en/ (last access: Dec. 30, 2015)
Installer available at: https://github.com/boulis/Castalia (last access: Dec. 30, 2015)

[22] Documentation, instructions and installation files at: https://omnetpp.org/ (last access: Dec. 30, 2015)

[23] Developed scripts, modified applications and additional files available at: http://www.jaumeguasch.com/tfm/index.html

Jaume Guasch is currently a MISTIC fellow at Open University of Catalonia (UOC). He obtained the Telecommunications Engineering degree from Technical University of Catalonia (UPC) in 1991 and an MBA from ESADE school in 2001. He has been working for several companies and organizations from private small companies to public government funded institutions from electronics systems design and manufacturing to research, innovation and technology transfer promotion and management.