EMMON - EMbedded MOnitoring
ARTEMIS JU – FP7

D7.11 NETWORK PLANNING AND ANALYSIS TOOLSET

EMMON

Agreement Ref.: *100036*

DISCLAIMER
ARTEMIS JU Contract Report
The work described in this report was performed under ARTEMIS JU contract. Responsibility for the contents resides in the author or organization that prepared it.

Date:	*2011-10-03*
Pages:	*26*
Status:	*Approved*
Dissemination Level:	*PU*
Reference:	*FP7-JU-EMMON-2011-DL-WP7-014-D7.11*
Version:	*1*

Customer:

D7.11 NETWORK PLANNING AND ANALYSIS TOOLSET

EMMON

Authors and Contributors			
Name	**Contact**	**Organization**	**Description**
Stefano Tennina	sota@isep.ipp.pt	ISEP	Author
Vincenzo Ciriello	vciriello.ext@sesm.it	SESM	Author
Pedro Braga	plbraga@criticalsoftware.com	CSW	Contributor

Dissemination Level

Public

Revision History				
Version	**Revision**	**Date**	**Description**	**Author**
1	1	2010-04-30	First Draft for internal review	Stefano Tennina
1	2	2010-05-31	Ready for Approval	Stefano Tennina, Vincenzo Ciriello, Pedro Braga
1	3	2010-10-03	Approved version according to the results of the Interim (M24) Technical Review Report, ref: ARTEMIS-ED-21-09 of 2011-07-19	Gareth May-Clement

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. Introduction

1.1 Objective

The main objective of this document is to provide an overview of the toolset that we devised and used for network planning and validation, encompassing analytical and simulation models, node programming and experimental data gathering. In particular, simulation, analytical and experimental results produced with the help of these tools will be compared against each other in Deliverables D4.3 and D4.6.

1.2 Scope

The EMMON project is composed of eight (8) Work-Packages:

WP1 – Project Management, Procedures and Communication;

WP2 – Exploitation, Dissemination and standardization;

WP3 – Study of user environment and definition of requirements and needs;

WP4 – Research activities on Protocols & Communication Systems;

WP5 – Definition of HW platforms and sensors;

WP6 – Research on Embedded Middleware;

WP7 – Implementation and System Integration;

WP8 – Operational Testing & Validations.

This is a new deliverable that was added to the list of deliverables for WP7, containing the details of the toolset from DEMMON1, to show work performed for the project but not defined for delivery in the Technical Annex [AD-1].

This document has one (1) release or issue planned, at instant T0+27 months during project execution (being T0 the project start date as per the contract).

1.3 Audience

The target audiences of this document are:

- ARTEMIS JU and the Commission Services;
- WSN research groups;
- Consortium participants;
- EMMON WP4, WP5, WP6, WP7 and WP8 participants.

1.4 Definitions and Acronyms

Table 1 presents the list of acronyms used throughout the present document.

Acronyms	Description
AD	Applicable Document
BO	Beacon Order
C&C	Command And Control station
CH	Cluster Head
DC	Duty Cycle

Acronyms	Description
GUI	Graphical User Interface
GW	Gateway
LSWSN	Large Scale Wireless Sensor Network
QoS	Quality of Service
RD	Reference Document
SBD	Square Based Deployment
SN	Sensor Node
SO	Superframe Order
TBD	Triangle Based Deployment
TDCS	Time Division Cluster Scheduling

Table 1 - Table of acronyms

1.5 Document Structure

Section 1, Introduction, presents a general description of the contents, pointing its goals, intended audience and document structure.

Section 2, Documents, presents the documents applicable to this document and referenced by this document.

Section 3, EMMON Project Overview, presents an overview of the EMMON project.

Section 4, Global Framework, outlines the global frame work of our defined EMMON toolset.

Section 5, Network Deployment Simulator, presents the simulator used to plan the network deployment. Its outputs are used to evaluate the system performance at the WSN Patch level by the tools described in Section 6, Performance Analyzer Tools.

Section 7, Remote Programming and Testing, briefly summarizes the tools used to program the nodes and to get from the network the experimental results to be further compared against simulation and analytical results that will be detailed in Deliverable D4.3.

Finally, Section 8, Conclusions, reports some concluding remarks.

Annex A, Network Deployment Tool GUI, presents some insights about the Graphical User Interface defined by our Network Deployment Tool.

Annex B, WSN Programming Tools, presents the UNIX Shell scripts used to program the network.

2. Documents

This section presents the list of applicable and reference documents as well as the documentation hierarchy this document is part of.

2.1 Applicable Documents

This section presents the list of the documents that are applicable to the present document. A document is considered applicable if it contains provisions that through reference in this document incorporate additional provisions to this document.

[AD-1] "Technical Annex", EMMON Project, ARTEMIS Joint Undertaking Call for proposals ARTEMIS-2008-1, Grant agreement no. 100036, 2010-01-31.

[AD-2] "D4.5 Specification of multilevel communication protocol", EMMON Project, FP7-JU-EMMON-2010-DL-WP4-005, Version 2, 2010-12-07.

2.2 Reference Documents

[RD-1] "ESA PSS-05-03 Guide to the software requirements definition phase", ESA, ISSN 0379-4059, Issue 1, May 1995.

[RD-2] P. Scheuermann. Sidnet-swans – a simulator and integrated development platform for sensor networks applications, 2010. (http://users.eecs.northwestern.edu/ocg474/SIDnet.html).

[RD-3] Weingartner, E. and vom Lehn, H. and Wehrle, K., "A Performance Comparison of Recent Network Simulators", IEEE International Conference on Communications, 2009. ICC '09.

[RD-4] Barr, Rimon and Haas, Zygmunt J. and van Renesse, Robbert, "JiST: an efficient approach to simulation using virtual machines: Research Articles", Softw. Pract. Exper., Vol 35, No 6, 2005.

[RD-5] Schoch, Elmar and Feiri, Michael and Kargl, Frank and Weber, Michael, "Simulation of ad hoc networks: ns-2 compared to JiST/SWANS", Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008

[RD-6] E. S. Biagioni and G. Sasaki, "Wireless sensor placement for reliable and efficient data collection", IEEE HICSS'03, pp. 127b, 2003.

[RD-7] K. Xu, G. Takahara and H. Hassanein, "On the robustness of grid-based deployment in wireless sensor networks", ACM IWCMC'06, pp. 1183 – 1188, July 2006.

[RD-8] Kenan Xu; Quanhong Wang; Hassanein, H.; Takahara, G.; , "Optimal wireless sensor networks (WSNs) deployment: minimum cost with lifetime constraint," *Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005), IEEE International Conference on* , vol.3, no., pp. 454- 461 Vol. 3, 22-24 Aug. 2005 doi: 10.1109/WIMOB.2005.1512937

[RD-9] Kenan Xu; Hassanein, H.; Takahara, G.; Quanhong Wang; , "Relay Node Deployment Strategies in Heterogeneous Wireless Sensor Networks," Mobile Computing, IEEE Transactions on , vol.9, no.2, pp.145-159, Feb. 2010 doi: 10.1109/TMC.2009.105

[RD-10] P. Jurcik, R. Severino, A. Koubaa, M. Alves, and E. Tovar. Dimensioning and worst-case analysis of cluster-tree sensor networks. ACM Transactions on Sensor Networks, 7(2), August 2010.

[RD-11] Open-zb: Open source toolset for ieee802.15.4 and zigbee, 2010. (http://www.open-zb.net/index.php).

[RD-12] Z-monitor: A monitoring tool for IEEE 802.15.4 wireless personal area networks, 2010. (http://www.z-monitor.org).

3. EMMON Project Overview

The EMMON project is an European Research and Development (R&D) project, sponsored by the 7th Framework Programme (7FP), ARTEMIS Joint Undertaking (JU) initiative and integrated in the Industrial Priority "Seamless connectivity and middleware".

EMMON motivation is originated from the increasing societal interest and vision for smart locations and ambient intelligent environments (smart cities, smart homes, smart public spaces, smart forests, etc). The development of embedded technology allowing for smart environments creation and scalable digital services that increase human quality of life.

The project goal is to perform advanced technological research on large scale distributed Wireless Sensor Networks, including research and technology development activities in order to achieve the following specific objectives:

- Research, development and testing of a functional prototype for large scale WSN deployments;
- Advance the number of devices by one order of magnitude, by real world validation (10 thousand to 100 thousand nodes);
- Advance the number of devices by two orders of magnitude, by simulation (100 thousand to 1 million nodes);
- Improve reliability, security and fault tolerance mechanisms in WSN;
- Identify and capture end-user needs and requirements, as well as operational constraints;
- Determine a path for exploitation of project results;

EMMON's main objective is the development of a functional prototype for the real-time monitoring of specific natural scenarios (related to urban quality of life, forest environment, civil protection, etc) using Wireless Sensor Network (WSN) devices. The goal of the project is to develop the technology to effectively monitor and control an area of 50 square km.

Areas of application for the project include a multitude of physical environments where continuous, large scale monitoring and situation analysis are of great interest, such as hydrographical systems (rivers and dam's), urban areas quality of life monitoring (pollution and noise), regional climate/marine monitoring, civil protection (forest fires, pollution propagation, etc), natural resources monitoring, energy production prediction, industrial plant monitoring, personal health monitoring and precision agriculture, just to name a few.

The increased environment awareness and detection of abnormal variations, allied with the possibility to rapidly broadcasting alarms and alerts, improves human quality of life and sustainability.

Project main results include:

- Large scale deployment of a fully-functional system prototype in a real world scenario (composed by thousands of nodes);
- New WSN embedded middleware with better overall energy efficiency, security and fault-tolerance;
- New efficient and low power consumption WSN multilevel communication protocols and reliable middleware for large scale monitoring;
- Simulation models for WSN behaviour analysis;

- Centralized C&C Centre for easy and centralized monitoring;
- Mobile C&C station or device for local access, diagnosing, viewing and troubleshooting of the network;

EMMON is structured on eight (8) work-packages (WP1 to WP8):

WP1 and WP2 include management, dissemination, exploitation and standardization activities;

WP3, WP4 and WP6 include the main RTD activities;

WP5, WP7 and WP8 aggregate all integration, implementation and testing activities.

Figure 1, illustrates the work-packages distribution within project areas and how they are related.

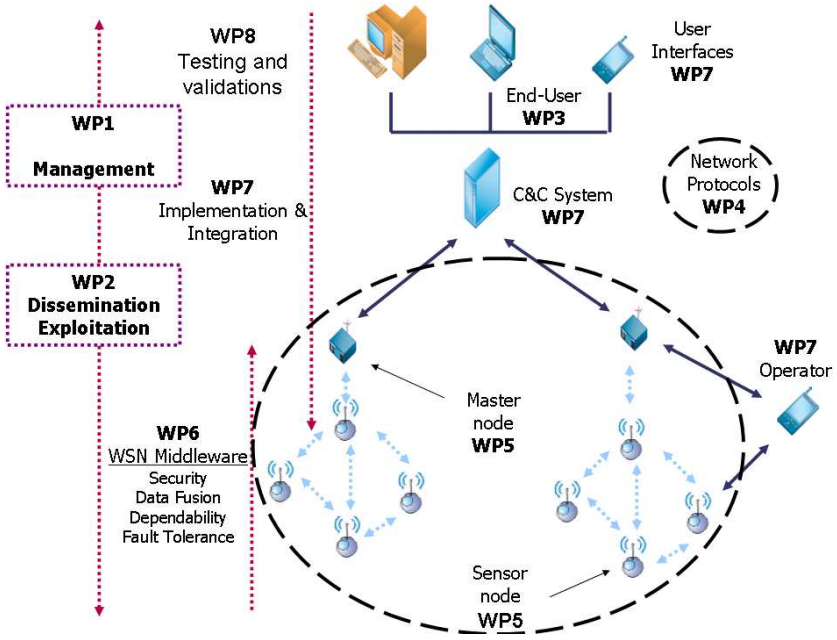

Figure 1 - EMMON system overview and work package decomposition

3.1 Work-Package 7 Overview

"WP7 - Implementation, System Integration" objective is to merge the end-users' requirements and implement all system, middleware and device interfaces required for the C&C module, converging all research and implementations into a common system prototype that can be used for system and module testing. The WP is split into six (6) Tasks:

"T7.1 - C&C requirements and interfaces design";

"T7.2 - C&C system and device Architecture Design";

"T7.3 - New interaction Methods";

"T7.4 - Implementation & Integration";

"T7.5 - WSN Networking systems implementation";

"T7.6 - Complexity Engine to interpret and forecast network signal propagation".

4. Global Framework

This section outlines the toolset that was designed for network dimensioning and analysis, nodes testing and programming, etc. Most of the results presented in the deliverable D4.3 (Simulation Results) derive from the use of these tools.

Figure 2 provides an overall perspective of the EMMON toolset, illustrating the integration between the different components (inputs/outputs) into a single framework. Each component is outlined in next sections.

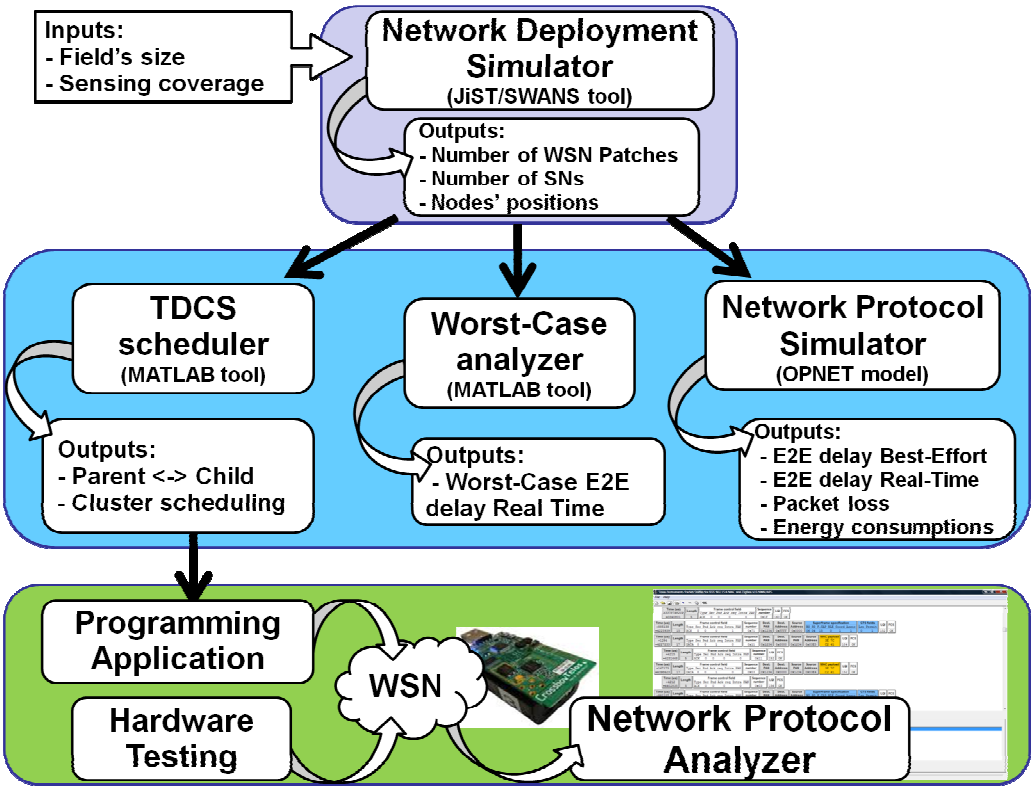

Figure 2 – EMMON toolset for network planning, dimensioning simulation analysis and test/programming.

5. Network Deployment Simulator

Starting from a set of defined target inputs, i.e., the size of the field to monitor and a desired level of coverage, the Network Deployment Simulation tool outputs how many GWs, CHs and SNs are needed to meet those requirements and where to put them, assuming the Cluster–Tree network model described in [AD-2]. Therefore, in EMMON it is assumed that we will have some control over node deployment: CHs and GWs are placed in order to maximize network connectivity, while SNs are generically supposed to be scattered all over the monitored area.

Simulation has been used to study the optimal deployment problem. We built upon an Open Source tool named SiDNet-SWANS [RD-2], built over JiST, a java based discrete event simulator. We choose JiST/SWANS because it allows simulating LSWSN easily. The results in [RD-3] confirm that JiST/SWANS presents better time execution performance, compared to more recent simulators as NS-3 and OmNet++. Moreover [RD-4] shows that this JiST/SWANS is really capable to simulate networks composed by a number of nodes very high (up to one million) in acceptable time and using machines equipped by common hardware. Currently, for NS-3 and OMNeT++ experiments showing similar capabilities are not available. Finally, the work [RD-5] ensures the qualitative features of the JiST's simulation, comparing it with the well known simulator NS-2 and getting very similar performance results.

The deployment problem in EMMON is composed by two levels:

I. Inter-patch deployment: how to distribute the WSN patches on the field;

II. Intra-patch deployment: how to place the nodes internally to the patch, in particular how to arrange the CHs and the GW;

The designed tool is based on innovative deployment strategies. The inter-patch deployment provides to organize the field as a grid, where each cell has a specific geometric shape (square, triangular and other shapes). Figure 3 shows three examples of grid: squares, triangle and hexagon.

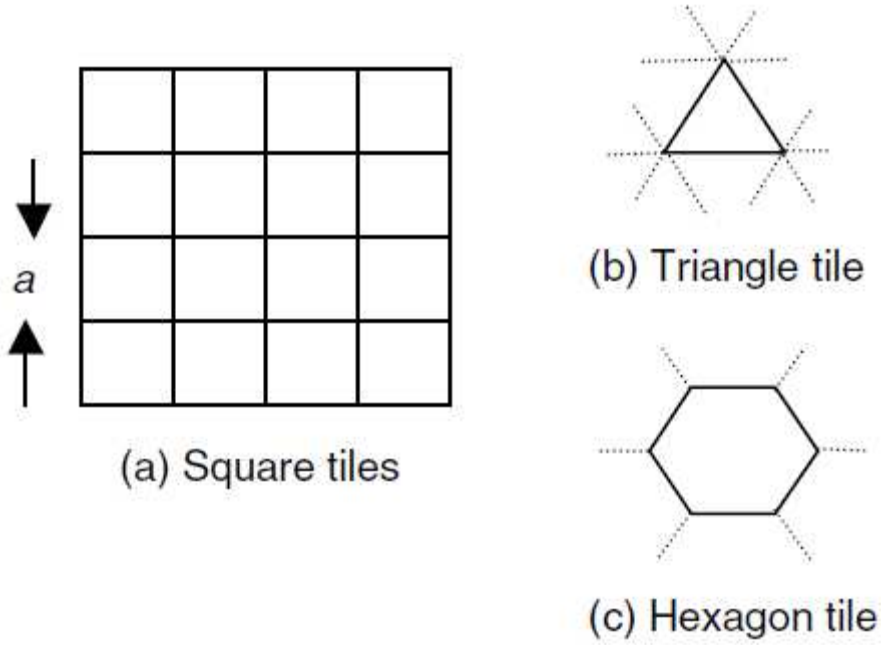

Figure 3 – Grid Deployment Examples

This deployment strategy has already been discussed in several studies in literature [RD-6], [RD-7], but we use it in a different way. In literature approach, the nodes (or a set of nodes in some cases) are placed in the vertex of each cells (e.g., in the corner of the square, or in the vertex of triangles). Whereas in our approach the nodes are placed internally to each cell: each of them contains a WSN patch, starting to the cell's centre, where is placed the GW. In particular, the designed tool provides two alternative grid deployments: *square based deployment* (SBD), where each WSN patch is circumscribed in a square and *triangle based deployment* (TBD) in which the patches are inscribed into equilateral triangles.

Related to the intra-patch deployment, the objective is to organize the WSN patch deployment as shown in Figure 4: the green points represent the CHs, while the red is the GW. Recall that the WSN Patch is logically organized according to a tree based topology in which the GW acts as the root and each WSN cluster as a node. Hence, the designed deployment provides two CHs "rings" that have been placed around the GW for any patch, while SNs are distributed in a random way within each WSN Patch and aggregated to one of WSN Cluster.

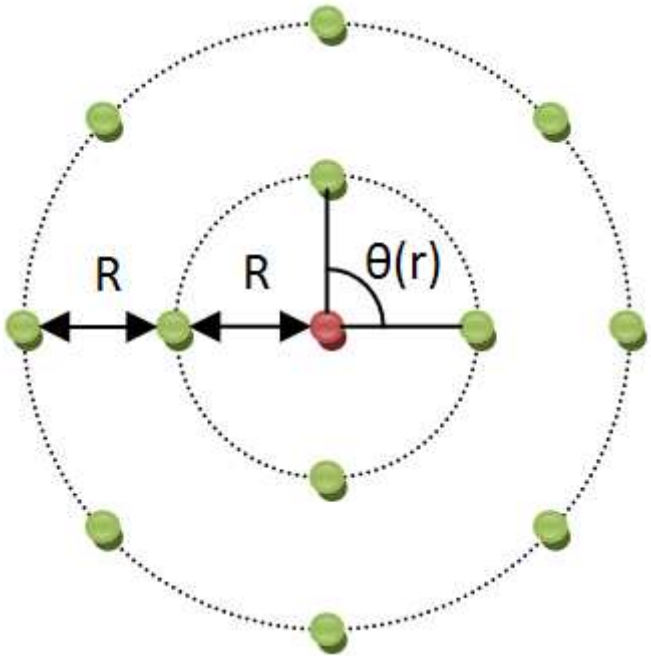

Figure 4 – Intra patch deployment

This deployment planning tool takes into account that a desired field coverage requirement must be achieved under realistic signal propagation conditions and that each resulting child should have at least two alternative candidate parents. In particular, this tool enables the comparison between the SBD and TBD strategies through the two following performance indices: (i) coverage and (ii) connectivity.

According to the definition reported in [RD-8] and [RD-9], a sensing field is considered to be completely covered if every point in the field (regular or irregular area) is within the sensing range of an active sensor. Regarding to the sensing range, each sensor node has a sensor reading range of X meters, which means it can acquire the temperature, light, humidity, as well as other physical quantities within a finite range surrounding it. In EMMON architecture, it is assumed that only the SN is able to perform sensing activities (the CHs and GWs in this way saving energy because they spend their energy to network forwarding activity).

Connectivity refers to how much of the generated data can ultimately arrive at the C&C. A WSN is considered to be fully connected if all nodes can deliver their data to the destination.

We are interested in checking that both properties are verified at the same time, therefore for field coverage we say that a cell is field covered if: there is at least one sensor less than X meters far away from its centre and that sensor can deliver the data to its GW.

Figure 5 reports an example of this concept: the triangles, circles and pentagon represent sensor nodes, cluster heads and gateway respectively. Supposing that each SN has a sensing range equal to one cell, e.g., each SN can sense the data by the eight cells adjacent to it only. The blue lines represent the link between nodes. The resultant analysis is the green cell represent field point that are field covered, because are covered (there is a SN in an adjacent cell at least) and the SN that cover the cell is connected (exist a path that connects it to the GW). The red cells aren't covered cells because they aren't adjacent cells where is placed a SN; while the violet cells are covered but aren't connected (the SN is connected to a cluster, but this CH don't have any one path to the sink).

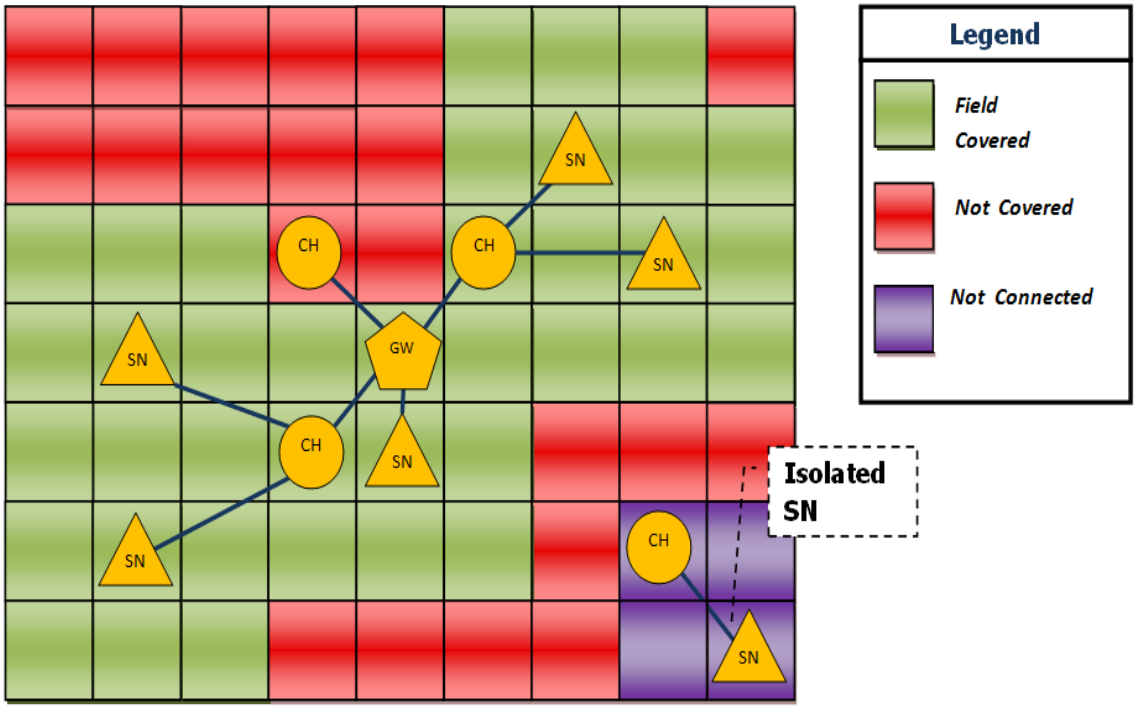

Figure 5 – Field coverage analysis example

The field coverage analysis algorithm is reported in Figure 6. It is an external tool, executed in post-simulation phase, which requires in input the simulation results and produces in output the field coverage analysis. In particular, the algorithm requires the nodes positions list Ls and considers the sensing range Rr as parametric. The field is represented as a Boolean matrix of dimensions where each element denotes if the corresponding cell is field covered. Basically, the algorithm provides that for each sensor in the nodes positions list is considered field covered all cells that are distanced by the sensor less than Rr. To this aim, we should consider as field covered any cell that falls in a circle which ray is equal to Rr and centred in the sensor node. Since that the representation of a circle on a square grid is complex, we adopt the following approach: rather than considering a circle, we consider a square which side is equal to Rr; we consider that a cell is field covered if this cell fall in this square and has a distance from the sensor less than Rr. The Field coverage is then calculated as the ratio between the covered cells and the whole field's area.

- **Input:**
  - L_s: list of sensor positions on the field.
  - R_r: sensor reading range.
- **Algorithm:**
  - Field is represented as a boolean matrix F.
  - $\forall(x, y) \in L_s$ considering a square A of side $2R_r$ centred in (x,y).
  - $\forall(x_i, y_j) \in L_s$:

$$if\left(\sqrt{(x-x_i)^2+(y-y_i)^2} < R_r\right) \quad => \quad F[x_i, y_j]=1$$

  - $\textbf{Field coverage} = \dfrac{\#\,\text{elements of F equal to 1}}{\#\,\text{field's point}}$

Figure 6 – Field coverage analysis algorithm

6. Performance Analyzer Tools

Focusing on the portion of the EMMON architecture below the GW, i.e., on a single WSN Patch, the outputs from the network deployment simulator are translated into appropriate inputs for three tools: TDCS Scheduler, Worst-Case Analyzer and Protocol Simulator. These tools are applied to each WSN Patch individually, as outlined next.

6.1 TDCS Scheduler

In the Cluster-Tree topology, beacons are messages sent by every local coordinator of any WSN Patch (i.e., the GW and the CHs) and serve to maintain the synchronization among the nodes of each cluster. This has the advantage of improving the coordination to save energy (reduce retransmissions, put the nodes to sleep and wake them up again in a synchronous fashion) and of guaranteeing a given level of QoS. However, to preserve the coordination and avoid intra–cluster collisions, the TDCS algorithm [RD-10] is needed. This mechanism involves the definition of the Start Time values of the MAC protocol, such that the active portions of each cluster are interleaved during the inactive portion of all the others sharing the same collision domain, as shown in Figure 7.

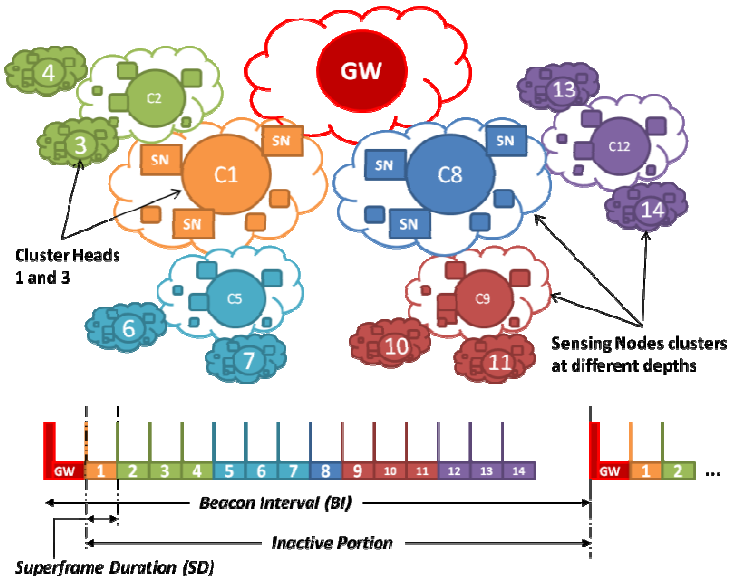

Figure 7 - Time Division Cluster Scheduling

This design choice leads to a given upper bound on the allowed number of clusters (Γ) and related duty cycles (DC). Knowing that 0 <= SO <= BO < 15, the maximum number of allowed clusters is $\Gamma=2^{BO-SO}$ and the duty cycle is $DC=1/\Gamma=2^{SO-BO}$. These relations are shown in Figure 8 and Figure 9. As a consequence, under the assumption that the couple (BO,SO) is the same for every cluster in a WSN Patch, at design time, those relations help identifying the best trade-off between Γ and the related DC.

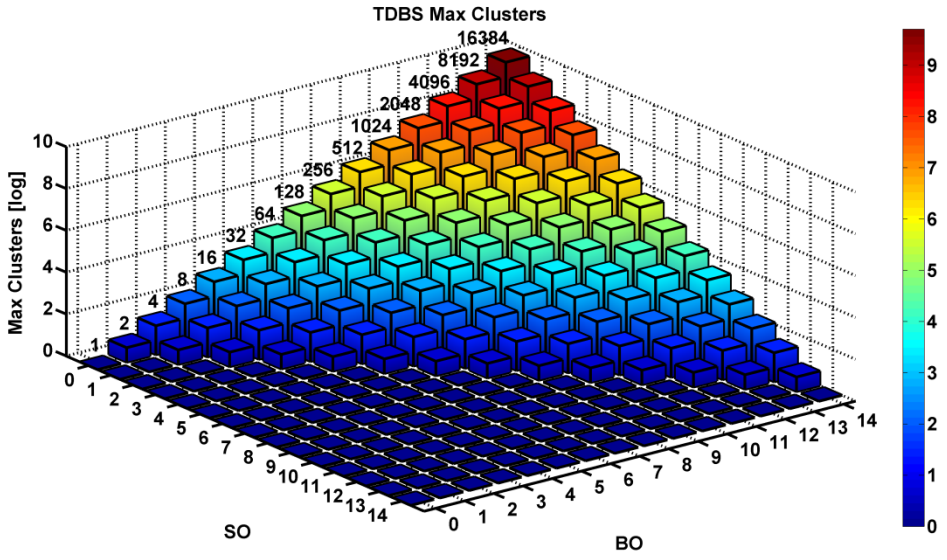

Figure 8 - Time Division Cluster Scheduling: upper bound on the allowed number of clusters

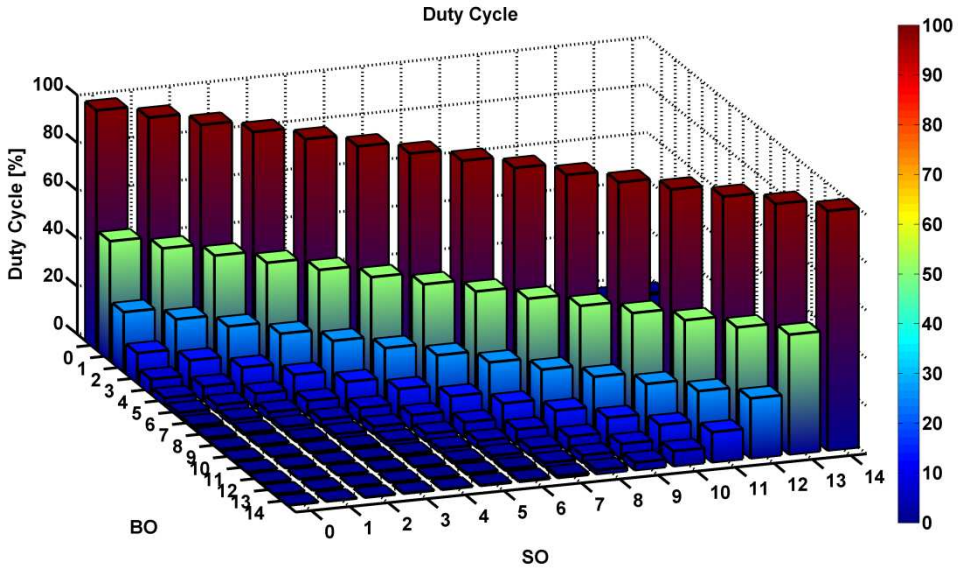

Figure 9 - Time Division Cluster Scheduling: duty cycles.

The TDCS Scheduler implements this mechanism. Given the WSN Patch characteristics (i.e., parent–child relation in the Cluster–Tree scheme), this tool built in MATLAB computes the minimum BO along with the start time for each cluster head to schedule its active portion (superframe) so that it does not overlap with the other clusters (as shown in Figure 7).

6.2 Worst-Case Analyzer

The Worst-Case Analyzer is a MATLAB tool (Figure 10) which estimates an upper bound for the end–to–end delay of the real–time traffic in IEEE 802.15.4/ZigBee Cluster–Tree Wireless Sensor Networks, i.e., the traffic whose packets are sent during the contention free portion (GTS slots) of the superframe.

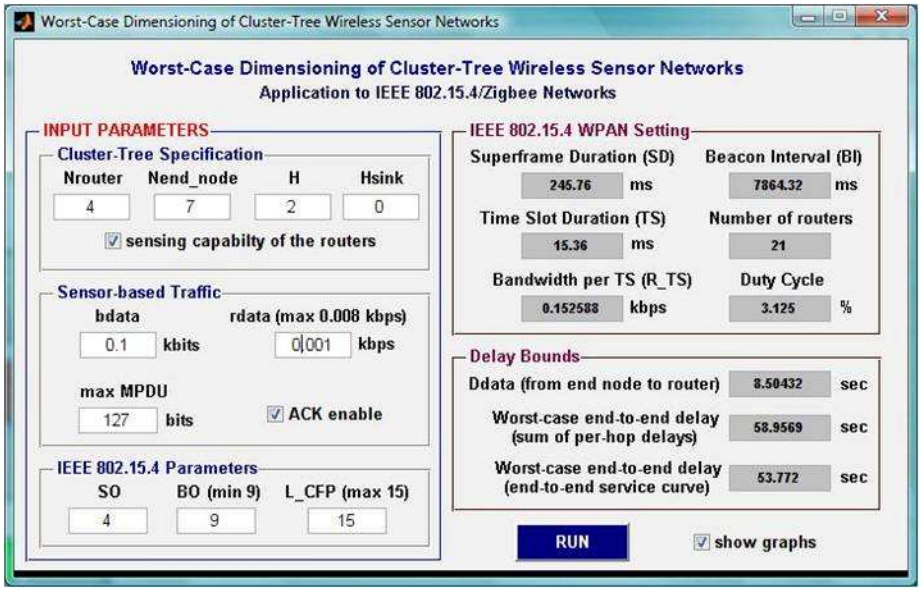

Figure 10 – MATLAB tool for worst-case analysis and dimensioning

This tool builds on the Network Calculus mathematical model, as described in [RD-10] and enables to iteratively find the best duty–cycle vs. delay/throughput trade-off, by varying network parameters such as sensor traffic.

6.3 Network Protocol Simulator

The performance of the communication protocol is evaluated using a simulator built in OPNET (Figure 11) [RD-11]. With the help of this simulator, we can infer end–to–end delays for both real–time and best–effort traffic, as well as compute network statistics such as packet loss, network throughput and lifetime, via per-node energy consumption estimation.

This simulator get inputs (network topology and nodes parameters) through an XML file which is automatically generated through an ad-hoc MATLAB script which takes as inputs the outputs of our Network Deployment Simulator (Section 5) and the windows offset from our TDCS scheduling tool (Section 6.1).

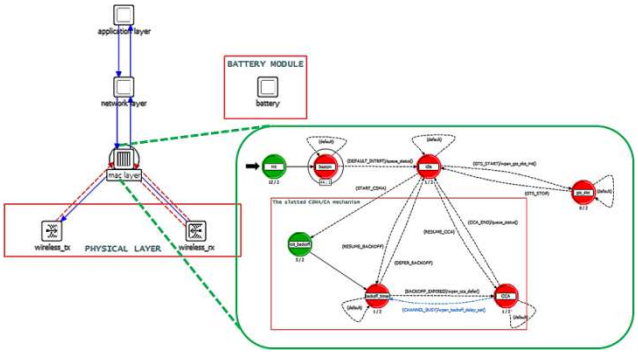

Figure 11 – The structure of a node within our OPNET simulation model

7. Remote Programming and Testing

Moving from the outputs of the TDCS tool, a set of appropriate MATLAB/Unix SHELL scripts are used to automatically generate header files (.h), compile the firmware of the nodes and load the compiled firmware in parallel into each node, through the USB tree (see Annex B). Interestingly, using this tool to program in parallel the nodes over the USB tree we were able to save 66% in programming time with respect to a classical serial programming. In particular, in the DEMMON1 deployment, we computed around 30 minutes to serially program a patch composed by 100 nodes, while with this tool the programming time reduced to no more than 10-12 minutes.

Finally, with the help of sniffer devices like [RD-12] and a custom–designed log parser, built in C++ and Matlab, EMMON specific data from the IEEE 802.15.4 frames are extracted. The outputs of the sniffer/parser (e.g., average and max delay) can then be compared with the theoretical (i.e., worst–case) and simulation (i.e., average and max) end–to–end delays.

We also designed an application to automate the testing of the hardware (USB cabling/hubs and TelosB nodes) having an element as reference (e.g., a previously tested TelosB node or USB hub). This testing tool work as comparison of results was fundamental to early identify: (i) 19 over 300+ TelosB with faulty humidity sensors (i.e., not usable as SNs); (ii) 2 out of 50+ USB hubs broken and (iii) a whole set of (3 m length) USB cables not compliant with the specification given to our local supplier.

8. Conclusions

In this deliverable we outlined the new development tools we designed and used to cope with network design, i.e. help designers to meet application requirements, from the network coverage to test, evaluation and validation of its performance. The toolset allows to efficiently design and thoroughly evaluate our EMMON large scale WSN based system.

The results of applying this toolset will be further illustrated in the deliverables D4.3 (Simulation Results) and D4.6 (Protocol Implementation) with a comparison of simulation, analytical and experimental outcomes.

Annex A

Network Deployment Tool GUI

The tool is provided by GUI, shown in Figure 12, which allows specifying all the parameters needed to EMMON's network planning. The parameters are stored in a configuration file. The GUI provides three panels:

- General configuration: contains the general parameters of the simulation. Note that is required to insert the number of SNs but isn't the number of CHs and GWs. This is because the SBD and TBD strategies computes automatically the number of GWs needed to cover the entire field; while the number of CHs depends by the configuration of the WSN Patch. The number of SNs, conversely, must insert by the network designer because it determines the resulting field coverage.

- WSN patch configuration: contains the parameters related to the number of CHs on the two rings NCH1 and NCH2; and the theoretical ring distance Rd.

- Deployment configuration: it allows choosing whether an SBD or a TBD strategy must be used, allowing the specification of all parameters needed by them.

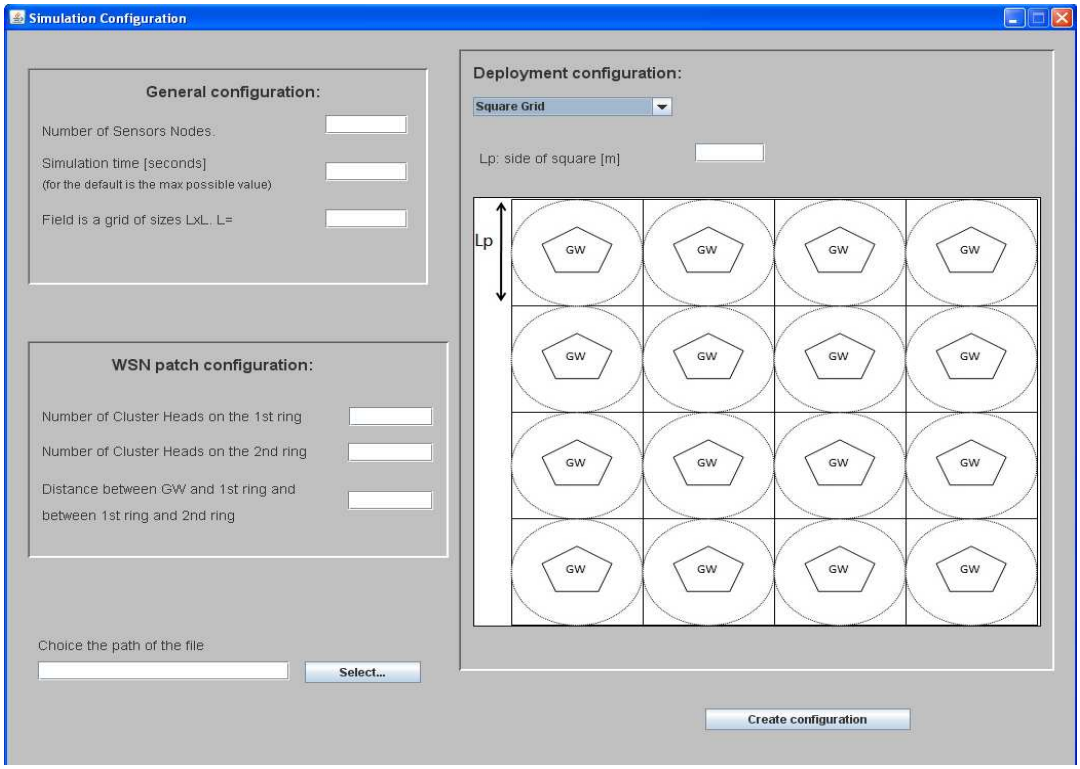

Figure 12 – Configuration tool related to SBD

Finally, the GUI allows specifying the path on which save the configuration file.

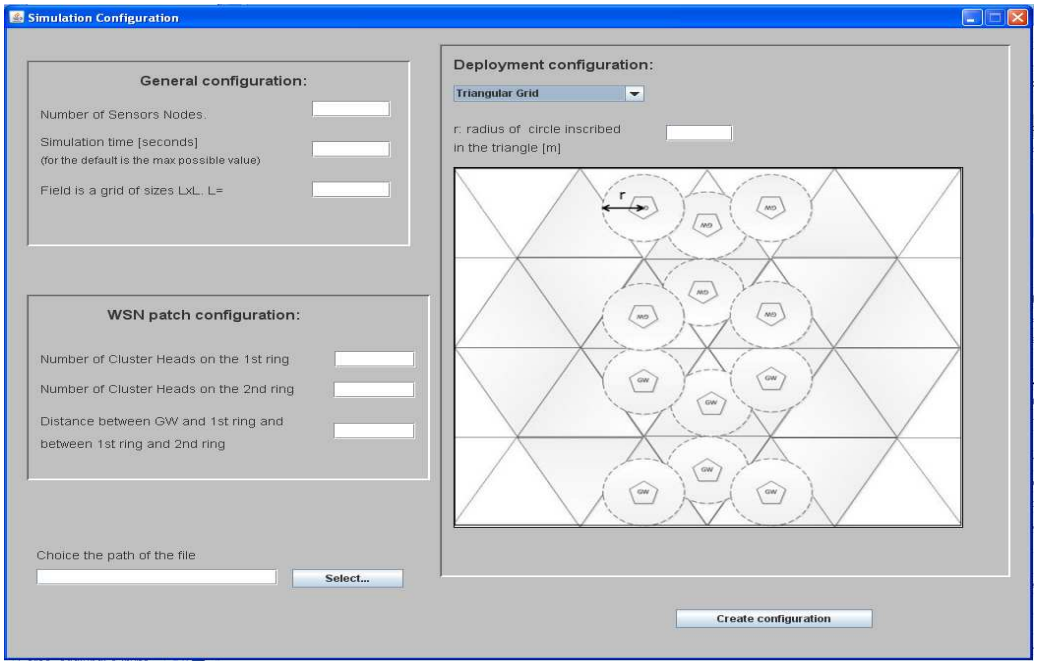

Figure 13 – Configuration tool related to TBD

This path must be provided in input to the simulation application.

The tools consist in three java applications, each of which uses as input the output provided by the previous tool:

- EMMONSimConfig: is the application related to the configuration tool GUI. It's a netbeans project. This application returns a file txt that must passed (by command line - view below) to the simulation application in jist-simulator. To run the tool simply:

Java EMMONSimConfig

- EMMONSimulation: the EMMON simulation application contained in SiDNET platform. It is an eclipse project. The project already contains all libraries needed to the working of simulator. The EMMON simulation application consists:

• sidnet.stack.users.sample_p2p.driver.driverAppCluster: it is simulation driver. It contains the parameters of the simulation as number of nodes, simulation time, etc and describes how the nodes must be placed on the field (contains the SBD and TBD algorithms).

• sidnet.stack.users.sample_p2p.app.AppCluster: it contains the node's behaviour. It contains the protocols that allow to the nodes to self-organizing according to the EMMON architecture. This includes:

- a heartbeat protocols that allow to each node to discover what are the nodes in its neighbours.

- an aggregation protocol, that allow to a SN to choice a cluster to which joining.

- a data collection protocol, that allows CH to retrieve the data by the SN.

- a routing protocol, that allows each CH to deliver the data to its GW.

This protocols are needed to check the connectivity parameter, e.g. to evaluate if the SN randomly deployed on the field is able to send the data to GW fallowing the EMMON architecture (cluster tree). The knowing of this protocols isn't essential, because they will not use in EMMON final prototype but it is used in simulation only to check the network connectivity.

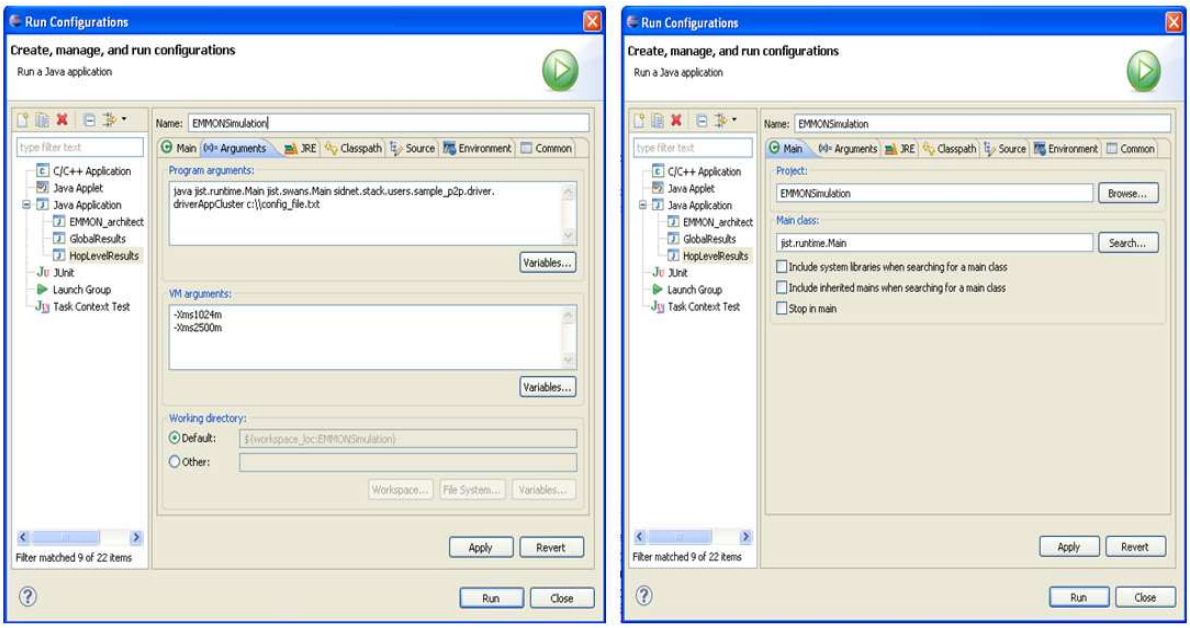

Figure 14 – ECLIPSE windows configuration

The ECLIPSE GUI is a simplest way to execute the application, but it is strongly recommended to execute the application by command line in the SO UBUNTU to performance reasons. For long simulation, is needed to growing the java heap memory size trough the jvm (java virtual machine) option: -Xms and –Xmx. This allows defining respectively the initial and the maximum size of heap memory. In Windows SO the maximum heap is 1.5GB, whereas in UBUNTU is 3GB around. To execute the application in UBUNTU SO fallowing this steps:

- Environment configuration: You must define the classpath in order to include the needed libraries. To this aim execute the script EMMON_config.sh.

- Change directory: opens trough shell the directory in which is contained the project and enters in the fallowing path \importedpackages\jist-swans-1.0.6\src

- Launch application: Put the fallowing command:

Java jist.runtime.Main jist.swans.Main sidnet.stack.users.sample_p2p. driver. driverAppCluster path_name

where the path_name is the path on which the configuration file is stored by the Configuration tool.

- Coverage analysis: it is the java application (eclipse project) that computes the field coverage. It requires a file named coverage_analisys.txt that is an output of EMMONSimulation.

Annex
B

WSN Programming Tools

To build the WSN software images for the motes, information on the network structure must be provided.

Network Structure Definition

The EMMON network structure used on DEMMON1 on December 2010 was split by three network configuration files, one for each patch.. Each patch of 100 motes was attached, via the USB tree, to the corresponding patch gateway PC. This way each gateway PC not only acted as an EMMON network component, but also as a TelosB programming environment to setup the network. We named those three network configuration files as "net_structure_p1_100", "net_structure_p2_100" and "net_structure_p3_100"; respectively for patch one, patch two and patch three.

These files were generated in two steps. First, a file only with positioning for each mote id was produced. This file associates each TelosB reference (tosid) to its coordinates (x, y, z). These positioning were determined and calculated using an absolute coordinate for the mote on the south-eastern corner of the room and the predefined distances between each node and this reference point. These distances were translated to absolute geodetic coordinates were computed by calculating the relation of linear meters in the room to longitude and latitude degrees. Moving one meter parallel to the main wall of the room would mean a change on the coordinates with two components: in the latitude and in the longitude.

Then a second step uses a MATLAB script to take this file as input and produce the corresponding "net_structure" file with all information per mote.
The format of this "net_structure" file is as follows: one line per mote. Each line has seven fields separated by spaces. Example:

```
r XBTJ2NOQ 9 2 1 86083703 411783300
r XBTJ4SZS 10 2 2 86083691 411783389
r XBTFECL3 11 2 2 86083679 411783479
e XBTJ4T2C 13 2 1 86083655 411783658
e XBTFJYIB 14 2 1 86083646 411783725
e XBTJ3H6C 15 2 1 86083917 411782592
(...)
```

The fields are detailed in Table 2:

#	Name	Description	Data format
1	Role	The role of the mote. Either a router (CH) or an end device (SN).	{"r" for cluster-head (router); "e" for sensor-node (end device)}
2	Tosid	TelosB reference (tosid) - the unique identifier of the node. Can be obtained out from the motes attached to the PC with "motelist" command.	Eight alphanumeric characters
3	Netid	The network ID.	Unsigned integer
4	Netdepth	The network depth of this mote; i.e., the number of nodes we must traverse in the tree up to the root node (EGW.TinyOS) which has depth 1.	Unsigned integer
5	netparent	The network ID (netid) of this mote's parent	Unsigned integer
6	pos_x	The "x" coordinates in EMMON units. Aligned with longitude.	Unsigned integer
7	pos_y	The "y" coordinates in EMMON units. Aligned with latitude.	Unsigned integer

Table 2 - Network structure fields definition

As an example, a file for a WSN with five motes can be handled manually: one EGW.TinyOS, one CH and three SN. In this case, because we do not specify the GW.TelosB on the Network Structure file, this file will have four lines as follows:

```
plbraga@demmon-gw:~/emmon/demmon1-v2/scripts$ cat net_structure_4-mote
r XBTIPPQX 1 1 0 86083420 411782730
e XBTFJYIB 2 2 1 86083380 411783871
e XBTJ3H6C 3 2 1 86083538 411782732
e XBTFJQO9 4 2 1 86083535 411782822
plbraga@demmon-gw:~/emmon/demmon1-v2/scripts$
```

Scripts to Build the Software and Program the motes

A script that we named "create-scripts.sh" generates three scripts: "del.sh", "compile.sh" and "program.sh", to erase all motes, to compile the binary images and to program the motes, respectively. This script will look for a file named "net_structure". Thus, this file must exist with this name in the scripts folder. This network file is the same as explained on 0. What we did was to create a symbolic link file with this name pointing to the correspondent structure file (e.g.: "net_structure_p3_100").

Here's the output of running this script:

```
plbraga@demmon-gw:~/emmon/demmon1-v2/scripts$ ./create-scripts.sh
Creating script to erase motes: del.sh
Creating scripts to Compile and to Re-program the motes: compile.sh and program.sh
plbraga@demmon-gw:~/emmon/demmon1-v2/scripts$
```

After this step has been performed, the scripts could be used several times to rebuild the binaries after a change in the implementation and to program the motes. The "create-scripts.sh" script only needs to be executed again if there is a change in the network structure; i.e., the steps on 0 are performed to produce a different "net_structure" file.