Alexander T.E.I of Thessaloniki
School of Technological Applications
Information Technology Department

Information Technology
Department

Bachelor Thesis

Research Methods on Development and Validation of Ad Hoc and Wireless Sensor Networks (WSNs)

Developed in collaboration with scientist members of
ICube Laboratory, University Of Strasbourg

By
Kosmas Kritsis
Student Registration Number: 05/2794

Thesis Supervisors
Dr. Periklis Chatzimisios
Dr. Antoine Gallais

Thessaloniki
July, 2015

Bachelor Thesis

Research Methods on Development and Validation of Ad Hoc and Wireless Sensor Networks (WSNs)

By
Kosmas Kritsis
Alexander T.E.I. of Thessaloniki

"Love thy neighbor as thyself, for it is only then that you can be useful to yourselves and of service to your fellow countrymen. There is nothing more worthwhile and rewarding in life than to work for the benefit of others. One can derive more pleasure from giving than from receiving."

Haile Selassie I

Αφιερωμένο στους γονείς μου Ευάγγελο και Ζαφειρούλα,

καθώς και στις αδερφές μου Άννα και Φωτεινή

Abstract

The verification of theoretical analysis is a vital step to the development of an application or a protocol for wireless networks. Most of proposals are evaluated through mathematical analysis followed by either simulation or experimental validation campaigns. Up to this point, we provide a detailed description of the development process and limitations of Wireless Sensor Networks (WSNs) as well as analyze a large set of statistics on articles published (i.e. 674 papers in total) in Ad-Hoc and WSN related top representative conferences over the period 2008-2013 (i.e. ACM/IEEE IPSN, ACM MobiCom, ACM MobiHoc and ACM SenSys). We mainly focus on the evaluation methodologies provided by researchers. More specifically, our goal is to explore the role of simulators and testbeds in the theoretical analysis of a scenario throughout the application development procedure. We show that there is a tendency that more and more researchers rely on custom or open testbeds in order to evaluate the performance of their proposals. Simulators indeed fail to reproduce actual environment conditions of the deployed systems. Experimentation with real hardware allows our research community to mind the gaps between simulation and real deployment. Still, as experimental approach through custom testbeds comprises a low reproducibility level (i.e., 16.5%), we investigate to what extent such performance evaluation methods will be able to bridge those gaps. We finally discuss experimental testbeds and their potential to replace simulators as the cornerstone of performance evaluation procedures.

Acknowledgments

This Thesis is a result of a research collaboration that took place during the academic year of 2013-2014 with the members of the ICube Laboratory at the University of Strasbourg, France. After a two-month period visit while participating at the Erasmus Internship program, I had the great opportunity to actively participate and obtain valuable knowledge concerning the structure and functionalities of a high-level federated WSN research laboratory, known as IoT Lab. Later we continued to cooperate by distance, with an outcome of a research article entitled "*Performance Evaluation Methods in Ad-Hoc and Wireless Sensor Networks: A Literature Study*" that was accepted on April 2015 for publication in the IEEE Communications Magazine.

First of all I would like to gratefully thank my supervisor, Dr. Periklis Chatzimisios, Associate Professor at the Department of Information Technology at Alexander T.E.I. of Thessaloniki, for his constant support though-out my BSc studies and confidence regarding my skills in participating in a research project, as well as his guidance in my future scientific career. Moreover, I would like to express my deep appreciation to my research supervisor Dr. Antoine Gallais, Associate Professor at the University of Strasbourg, for the given opportunity to visit the ICube laboratory and participate in a research study, in addition to his guidance, support, understanding and patience during our collaboration.

Special thanks go also to my advisor, collaborator and friend Georgios Z. Papadopoulos, who is currently a PhD candidate at the University of Strasbourg, France.

Moreover, one person, E.K., motivated me during the last 3 years to succeed my goals, by showing respect and patience to our relationship.

Last but not least, I would like to deeply express my wholehearted love to my parents and sisters for encouraging and believing in me, because nothing would be possible without their support and help.

Table of Contents

1. Introduction

1.1 Motivation and Outline

Computer networks can be considered as one of the greatest achievements of humanity, since they enable the users to communicate almost instantly, regardless of their location. However, after many decades of research, wired networks have reached maturity, thus evolving to new wireless technologies that further introduce a great diversity of possible applications, such as Ad Hoc and Wireless Sensor Networks (WSNs). These networking technologies emerged in order to provide solutions for different scientific problems, including health care, disaster recovery, environmental monitoring as well as smart cities and the modern Internet-of-Things (IoT) applications.

The primary purpose of this B.Sc. Thesis is to provide a detailed overview considering the development lifecycle of Ad Hoc and WSN systems, and in particular to analyze the different performance evaluation methods, which are employed by the research community. Therefore, the Thesis is organized as follows. In Chapter 2, we introduce a brief description of a typical WSN research lifecycle along with the involved procedures. Next, Chapter 3 provides a detailed description of the simulation requirements followed by a brief overview of the available WSN simulation tools. Similarly, Chapter 4 describes the basic requirements and structures of the experimental testbeds, as well as it summarizes the different WSN hardware and experimental laboratories. Finally, after a thorough study of 674 scientific articles, Chapter 5 analyzes various statistics concerning the current trend of validation methodologies in the research field of WSN and Ad Hoc networks.

1.2 Computer Networking

A computer network can be defined as the connection between two or more devices, over a common communication channel, in order to share data and resources. This definition can be very comprehensive, however its simplicity obscures the great diversity of possible network models and technology utilization such to achieve networking. In the modern era, people in their everyday life rely on networks without understanding the complexity of the

involved technology. This demand motivated a rapid evolution in network technologies, from wired communications to contemporary innovative wireless models. Despite the underlying architecture of networks, a broader classification can be applied based on the area coverage and transmission medium utilization. A brief description of the different types of networks can be introduced as follows.

i. Local Area Network (LAN)

Typically, a LAN is used to connect computers at a single area such as homes and small offices, where the users need to communicate amongst them and not with the outer world. A single person is usually the administrator and manages the network, which can be wired hubs and switches and/ or wireless access points.

ii. Wireless Local Area Network (WLAN)

A WLAN is a LAN that is based only on wireless connectivity (usually by employing IEEE 802.11 protocols). The users by employing wireless interfaces are able to exchange information through an access point. The air is considered to be the transmission medium. Therefore, every network interface is equipped with an antenna, which produces radio signals in order to be able to participate in the WLAN.

iii. Metropolitan Area Network (MAN)

A metropolitan network covers bigger area compering to LAN. A common MAN interconnects individual LANs within a city. Hence, a MAN should support routing services.

iv. Wide Area Network (WAN)

A WAN interconnects users within large areas, like countries or even the entire world (Internet). Separate LANs and MANs are interconnected in order to form a WAN, by utilizing router devices.

v. Campus Area Network (CAN)

Computer networks that operate over university campuses or big corporation offices are referred as campus area networks. Following similar principles to MAN, individual LANs are connected in order to implement a CAN. However, the occupied area is smaller than MANs.

vi. Storage Area Network (SAN)

A storage network is used to connect servers with storage devices by utilizing Fiber technology in order to achieve high bandwidth. SAN improves storage efficiency for storage-oriented applications.

vii. System or Cluster Area Network (S/CAN)

A system or cluster area network interconnects high performance computers over fiber channels so as to provide high bandwidth. This class of networks is used in distributed computing for cluster configuration.

viii. Personal Area Network (PAN)

A PAN is the smallest network in terms of area coverage. The personal networks emerged in order to interconnect devices around a single person. Such networks typically involve PCs, telephones, Personal Digital Assistants (PDA), printers and so on, which communicate in a way to provide optimized services according to the requirements of the individual. Wired PANs may communicate over USB or FireWire interface, while wireless PANs employ Bluetooth, or IEEE 802.15 radio interfaces.

1.3 Wireless Communications

In the last two decades, wireless communications emerged as a result to the increasing utilization of Information Technology (IT) systems by the people, thus becoming an integral part of several types of communication devices, as it allows users to communicate even from remote areas. Wireless technology involves the transmission of information over a distance without wires, cables or any other form of electrical conductors.

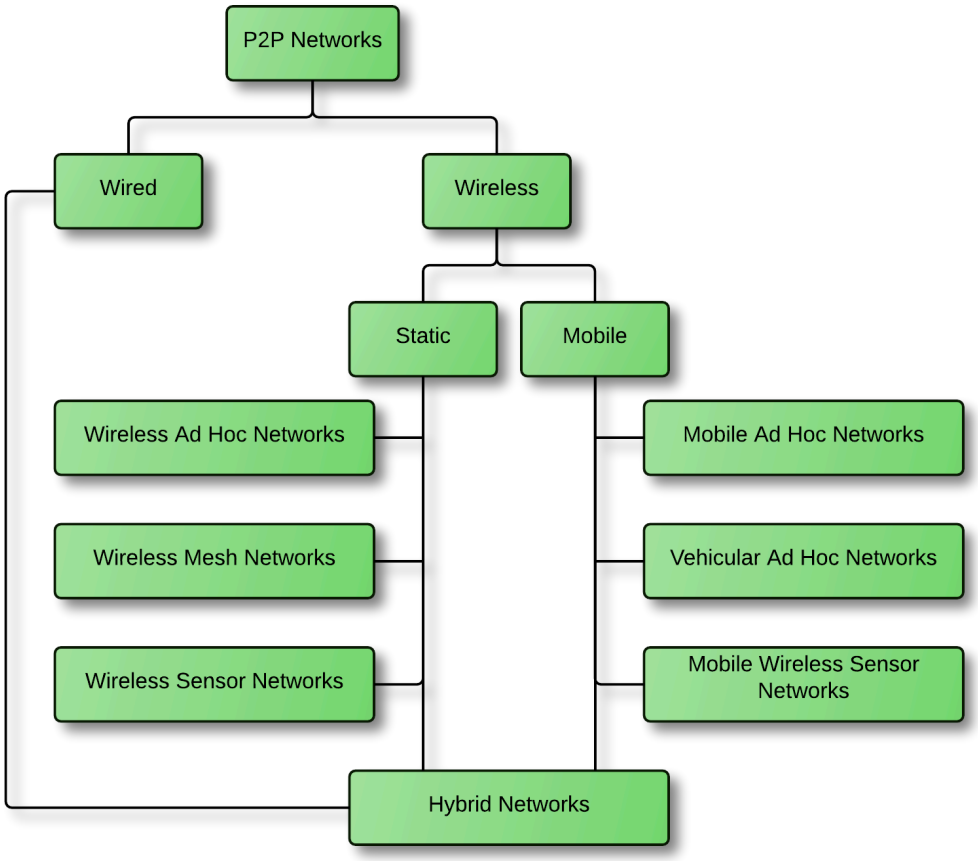

Figure 1. Wireless Networks Hierarchy

The transmitted distance varies from few meters to thousands of kilometers while providing freedom of movement and the ability to extend applications to different areas. Therefore many types of wireless networks and systems exist as presented in Figure 1, mostly by interconnecting various computer devices such as servers, PCs, laptops, smartphones, RFIDs, printers etc. Wireless networks can be installed conveniently while providing much more flexible and cost-effective solutions compared to traditional wired networks. However, our study will focus only on the wireless Ad Hoc networks and WSNs.

1.4 Ad-Hoc Networks and WSNs: Similarities and Differences

Ad Hoc Networks and WSNs are similar because both types can be considered as decentralized and distributed wireless networks while not requiring a significant network infrastructure in place. Two nodes of the network are able to communicate either directly (single-hop routing) or by involving intermediate relay nodes (multi-hop routing). Additionally, both Ad

Hoc and WSN nodes are typically powered over batteries thus requiring energy aware mechanisms in order to minimize the power consumption. Moreover, these networks communicate over an unlicensed radio spectrum and therefore they are vulnerable in interference by other radio technologies that operate in the same wireless band. Finally, both networks should support self-organization methods due to their distributed nature.

However Ad Hoc networks were developed during 70´s by the scientist in order to be employed by the US military. As to the present day there are various commercial applications based on Ad Hoc technology, which is quite different from the previously developed military systems and therefore they require a new approach to the problem (Gerla, 2005). The military solutions are mostly developed for a single purpose, thus employing unique hardware and software solutions in addition to their high development cost, which can not be adopted by the commercial applications.

Despite the aforementioned similarities, there are various fundamental differences, which most of them derive from their different nature. Ad Hoc networks were developed to interact closely with the user, since most of the nodes are devices used by human beings including laptops, PCs, PDAs, etc. On the other hand, WSNs do not focus on servicing the user, rather than interacting with the environment. Indeed, the nodes involved in a WSN are usually embedded devices that sense various environmental events and possibly actuate on their occurrence. Additionally, the number of nodes involved in a typical WSN scenario varies from tens to thousands of nodes, which introduces further density as well as scalability issues that are not required in a simple Ad Hoc network.

2. Research Process in WSNs

A WSN system is a combination of software, network and embedded engineering. These fields are well defined and therefore, WSN developers should be aware of the currently employed technologies and methods in the previously mentioned domains, so as to efficiently design new solutions. However, some of these practices should be modified in order to fit the specifications of WSNs. As far as it concerns the development lifecycle of applications that incorporate WSN technology, there are various methodologies that can be adapted. The concepts of Service-Oriented Architecture (SOA), agile development methods and networking practices of Mobile Ad Hoc Networks (MANET) and Peer-to-Peer (P2P) domains can be combined so as to establish an effective development framework for WSNs.

In a SOA-based system, the application is a logical set consisting of different software that incorporates in a way to perform certain tasks (Papazoglou, 2003). Furthermore, SOA is a popular method that is broadly employed in the design process of Web Services (Schroth & Janner, 2007). Sensornets act in a similar way, by categorizing the involved nodes into groups according to the services that they provide and thus enabling researchers to adapt popular Internet technologies and protocols in their designs. Moreover, SOA methods have to be modified so as to include the complex services of WSNs such as storage, routing and sensor readings. However, this system architecture may utilize additional component-based models for more detailed designs. The main difference between these two approaches is that components define the actual functionality of the system, while services are used to describe the logic and the interactions amongst the components (Petritsch, 2006).

On the other hand, WSNs are employed in a broad area of applications and therefore different parameters should be optimized accordingly. For instance, the data propagation time, the fault-resilience requirements, the network size, node mobility, code maintainability and re-usability are some parameters that differ in various scenarios. However, it is rational that improvement of some parameters affects negatively others. Balancing the major parameters of the system is a complex task and usually leads to iterative procedures between the designing and development phases of the project. Additionally, the project requirements often change from the initial ones during its lifecycle.

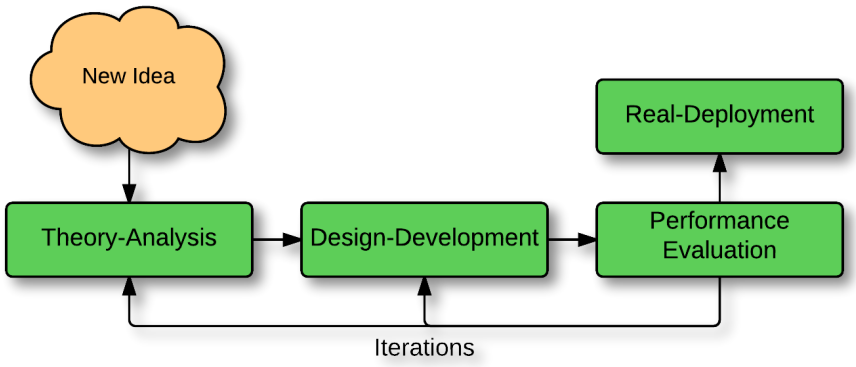

Figure 2. Typical WSN Project Lifecycle

Hence, agile methodologies are appropriate for developing WSN applications. This type of development can be identified due to the short and iterative development cycle as well as the constant and direct interactions between the members of the development team (Martin, 2003).

However, most WSN specific research projects follow relatively the same project lifecycle stages, which are the analysis phase, the designing-development phase, the performance evaluation phase and finally the real deployment phase, as presented in Figure 2. The following Sections in this Chapter are organized according to the aforementioned phases and further provide a brief description of the involved tasks as follows.

2.1 Theory – Analysis

Development process begins with the conception of the subjected problem. Researchers and engineers from various institutional or commercial departments cooperate in order to analyze and define the building blocks of the solution by collecting project requirements.

Early stage analysis of the sensor networks requirements is a critical task, due to the difficulties of accessing and maintaining the WSN system later to the post-deployment phase. The unique characteristics of WSN applications present a diversity of specific challenges, which enables researchers to apply various formal methods for system specification, verification and synthesis. In order to apply this rich set of methods and theories, the very first task is to build a formal description of the sensor network. Usually, a formal system description requires to be defined over high-level mathematical models, which can later be used for a variety of system analysis tasks, including simulation,

verification and performance evaluation (Dong et al., 2008). Moreover, the formalization of the WSN description allows convenient and faster development of WSN systems, as well as it enables a partial automation of this process (Meshkova et al., 2008).

Also the authors of (Meshkova et al., 2008) suggest a basic collection of parameters that need to be practically considered for any WSN system (Figure 3). These parameters can be further classified into four groups, according to the abstraction-levels of the development process that they are involved. Firstly, the main parameters specify the overall system performance and functionality, thus being the fundamental and the most important requirements of the project. Secondly, the network parameters describe the wireless network and its behavior according to the interactions between the nodes of the system. Next, the service parameters define the appropriate services and their behavior in conjunction with their inputs and outputs. Finally, the hardware and software requirements describe the devices and the employed software that collaborate in order to form the actual WSN system. Typically, these parameters are valuable during the designing and development stages of the project. A brief description of the aforementioned groups is introduced as follows.

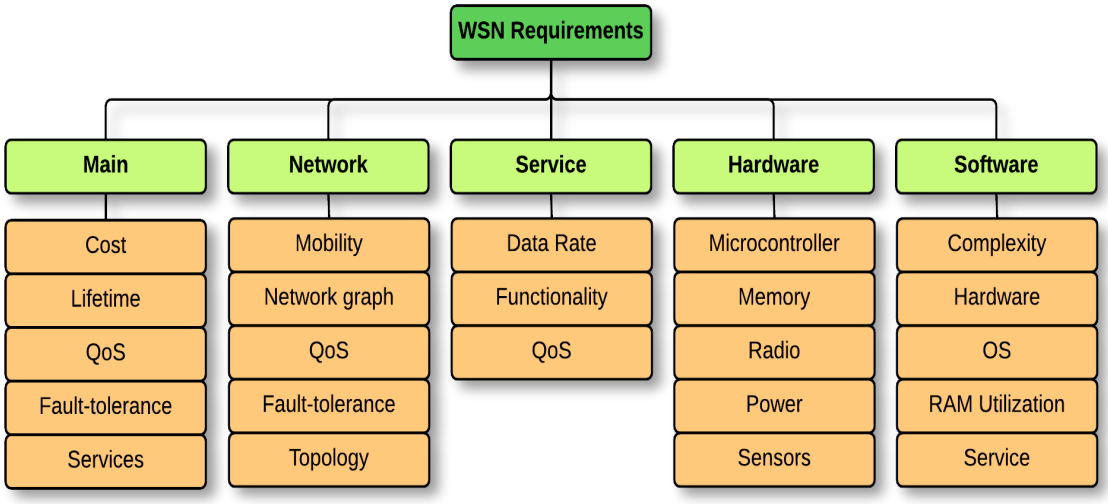

Figure 3. Typical WSN System Requirements (Meshkova et al., 2008)

2.1.1 Main Requirements

The main requirements consist of five basic parameters, which are the project cost, the network lifetime, the propagation delay, the fault-resilience and the services that specify the overall WSN functionality.

The total cost of the project is a combination of the hardware, the development and the deployment costs. The deployment cost also can increase in heterogeneous systems that employ individually specific nodes. Conversely, homogenous solutions are much cheaper due to their ability of self-organizing according to the overall system conditions. Nevertheless, in order to achieve this functionality, more effort is required during development and thus increasing the cost of this phase. Dynamic networks are more complex, however they provide a flexible structure that can self-adjust to various scenarios. Some examples of networking technologies that enable dynamic configuration are middleware frameworks in conjunction with code updates through the Erasable Programmable Read-Only Memory (EPROM) (Brown & Sreenan, 2013). On the other hand, static networks requires extensive designing efforts and test implementations in order to provide a convenient solution with minimal interference letter to real deployment.

The network lifetime is a major requirement because the network is functional only when it is considered alive. In other words, the network lifetime depends on the individual lifetime of the involved nodes. As a measurement for energy consumption, it forms the upper bound for the utilization of the sensor network resources. The lifetime of a sensor node basically depends on how much energy it consumes over time in conjunction with the available energy for use (Dietrich & Dressler, 2009). Typically the motes are powered over batteries and therefore they have limited lifetime. However, there are various available technologies that can be employed in order to further increase the nodes' lifetime. For instance, computational balancing can be applied by addressing parts of the processing load to the gateway or even to the user device. Although it has to be ensured that the gain from processing information on the gateway is greater than the communication cost of transferring data to it. Hence, this approach is suitable for small-scale networks. Additionally, energy harvesters are usually employed in order to extend the nodes' energy resource, as depicted in Figure 4 according to the authors of (Merrett et al., 2009).

As far it concerns the Quality-of-Service (QoS) of the project, two major parameters need to be considered, which are the propagation delay and the fault-resilience.

The maximum allowable propagation delay between the gateway and the farthest node depends on the subjected scenario. For instance, health care and emergency control are some applications that are delay-sensitive, thus requiring strict synchronization between the nodes. Additionally, delay-monitoring techniques are employed such to detect abnormal delays in the system (Zeng et al., 2009). Then the appropriate corrections must be applied so as to maintain the normal functionality of the network.

Fault-tolerance can be considered as a parameter in all the involved models, including the hardware and software of the system along with the underlying node and network models. Furthermore, fault-resilience mechanisms manage and specify the tolerance of the sensor network to situations such as node failures, outdated and imprecise information in addition to data losses and malicious data injections. For instance, security add-ons may be employed on top of back-up and relay nodes. However, these methods introduce additional computational procedures and thus reducing the network lifetime. Moreover, this approach requires extra development effort and deployable devices that further increase the overall project cost.

Finally, the services parameters specify the expected functionalities of the system, according to the user expectations and the purpose of its development. However, the initial requirements are based on assumptions that need further updating later on the development and validation phases of the project.

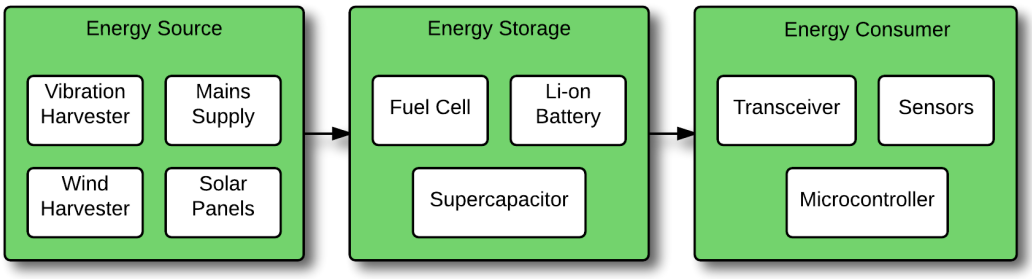

Figure 4. Typical Energy Components in WSNs (Merrett et al., 2009)

2.1.2 Network Requirements

The network requirements are considered to be critical because they describe the network where the application is about to be deployed. These parameters include information concerning the node mobility, the fault-tolerance, the appropriate bandwidth, the number of employed nodes, the geographical network coverage and the network symmetry by means of heterogeneity. The user can identify most network parameters, such as the expected bandwidth and node mobility, during the initial stage of conception. However, most of them are specified and adapted by the development team throughout the project lifecycle and thus affecting other requirements that need further adjustment. For instance, fluctuations in the geographical positions of the nodes influence the network coverage and heterogeneity requirements.

Globally the network is modeled and organized based on the graph theory as:

$$G = (V, E) \text{ with}$$

$$V = (\sum_{\iota=1}^{\iota=\kappa} \{n_i\}) = \{n_1, n_2, \ \dots \ , n_k\} \text{ and}$$

$$E = \left(\sum \{n_i n_j\}\right) \quad \forall \ n_i, n_j \in V \qquad \left(n_i \neq n_j\right)$$

The graph consists of a set of vertices V that represent the nodes of the network, in accordance to a set of edges E that further contains numerous pairs of nodes as subsets of V, which represent the links between the nodes. An edge $e = \left(n_i n_j\right) \in E$ exists if and only if n_i is in the transmission range of n_j and vice versa (Dimokas et al., 2007).

Initially, general network parameters such as the network heterogeneity, density and scalability, must be estimated and translated into a graph. Later in the development phase, designing options are limited and thus more detailed graph representations of the network can be composed and evaluated. However, most WSN systems are developed by highly qualified small teams and therefore these processes are typically merged.

2.1.3 Service Requirements

Similarly to SOA, during the conception phase, WSN services are described based on high-level abstractions that provide generic design solutions. Furthermore these solutions aim on providing further formalization of the WSN development process, and thus they are suitable for systems that employ heterogeneous designs.

Most WSN applications require various services in order to provide their functionality. For instance data aggregation, processing and decoding are crucial services performed by every node in the system. However in dense networks, such requirements can be satisfied by deploying special nodes that work as distributed service providers. Furthermore, the trade-off in service composition is to assign each required service to the appropriate service provider based on certain parameters such as the load balance, the propagation delay and the available network resources (Wang et al., 2009).

Typically, the service requirements include all the expected functionalities and services of the system. Also each service should consider the expected influence on the nodes' lifetime and the introduced information overhead in the network traffic.

2.1.4 Software and Hardware Requirements

The software and hardware requirements are valuable during the designing and development phases. The software parameters specify the required Operating System (OS), the minimum memory capacity of Radom Access Memory (RAM) and EPROM, as well as a list of software modules that are essential for the overall node and network functionalities. The hardware parameters specify the WSN hardware platform and its underlying components, including the type of sensors, memory, microcontroller chip, communication interfaces, radio and finally the available energy resources.

2.2 Design – Development

After the initial stage of gathering and analyzing the basic requirements of the subjected WSN solution, the researchers continue by investigating for available proposals that can be further adopted so as to minimize the development costs. However, in most cases the developers need to design new solutions so as to satisfy the requirement trade-offs between cost,

performance, network lifetime and QoS. Therefore, the initial task of the designing phase is to specify the appropriate network architecture. Next, the developers have to design the application by modeling the underlying protocol stack for the nodes of the network. Finally, at the development phase of the project, the abstract designing models are implemented into real code. A brief description of the procedures involved in the designing and development phases can be found as follows.

2.2.1 Network Designing Procedure

Through the network designing stage the developers have to decide on the corresponding network architecture that matches to the requirements of the project. As it is illustrated in Figure 5, the WSN network structure can be classified into four types, which are the single-hop star topology, the multi-hop mesh and grid topologies as well as the two-tier hierarchical clustered topology (Chen et al., 2009). The researchers should choose between these network architectures according to the scale of the network, by means of geographical latitude, as well as the behavior of the nodes such as mobility and fault resilience.

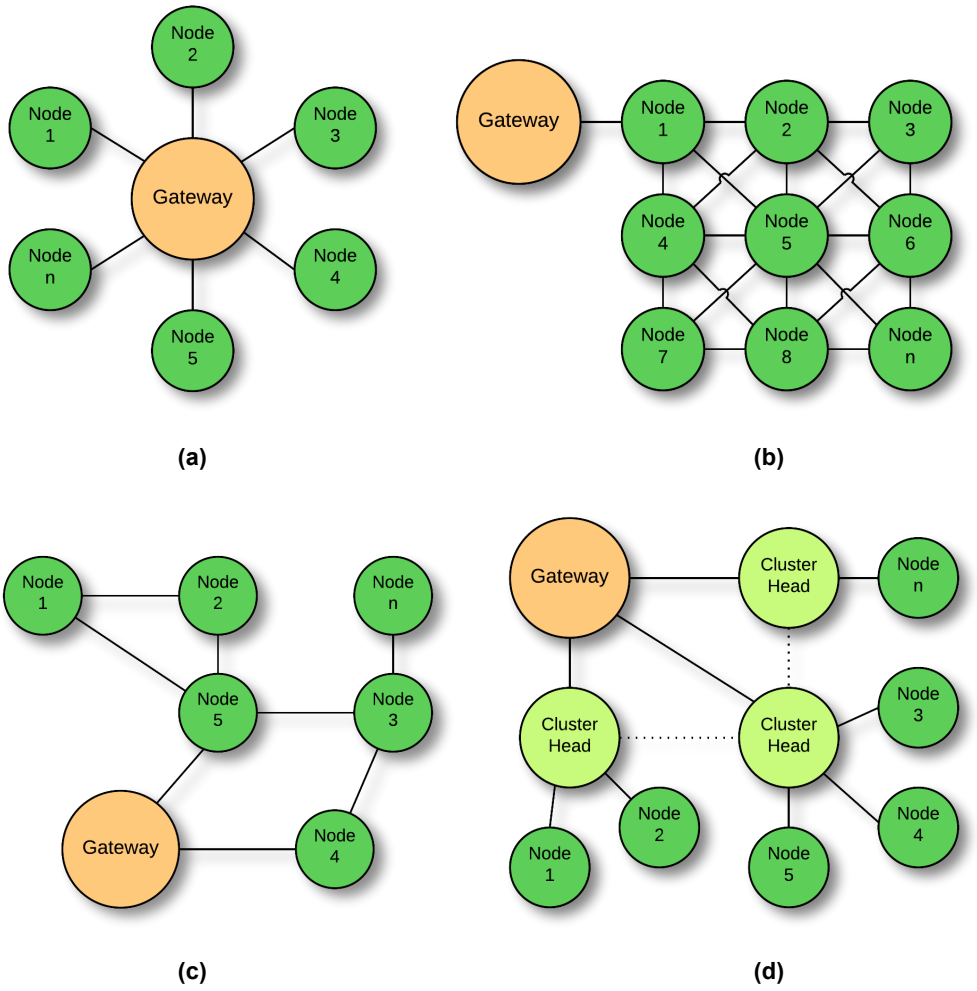

Figure 5. Typical WSN network topologies - (a) single-hop star, (b) multi-hop grid, (c) multi-hop mesh, (d) two-tier clustered

i. Single-hop star topology

A single-hop star topology is the simplest WSN network structure, which consists of the nodes that are directly connected with the gateway, as presented in Figure 5.a. This type of networking enables handy designing and minimizes the overall project cost due to its simplified modeling abstractions. Hence, it is a suitable solution for small-scale networks were cost and fault-resilience are often major concerns. For instance in medical applications the nodes should be free of deficiencies because even a single failure may lead to losses in the patient's state. Cost is important in scenarios such as smart home applications, where most of the nodes have the same sensory

capabilities. Furthermore, in small-scale networks the nodes can transfer computational load on the gateway device with low communication cost.

ii. Multi-hop mesh and grid topologies

For medium and large-scale networks there is a need to specify a routing process between the nodes of the network in order to transfer sensory information to the gateway. Therefore, each node relies to its neighbors to promote its data to the central sink or gateway, by adapting popular client-server and P2P approaches (Chawathe et al., 2003). Furthermore, according to the type of deployment, the developers choose between mesh and grid topologies. Grid topology is suitable for scenarios where the nodes follow a structured deployment, as presented in Figure 5.b. On the other hand, a mesh topology is optimum in applications where the nodes are randomly deployed, similarly to Figure 5.c.

iii. Two-tier hierarchical clustered topology

Nevertheless, the most popular network model for large-scale networks is the two-tier hierarchically clustered topology. It is a complex and role-specific architecture where nodes perform different functionalities in the network. According to this topology the nodes of a specific field transmit their data to the cluster head node of their area, which further promote their data either to the central gateway or to other cluster heads from neighbor regions, similarly to Figure 5.d. By employing nodes with different capabilities it can significantly improve the network performance. On the other head, the complexity of developing heterogeneous networks increases the overall project cost. Typically, the cluster head nodes are designed to be more powerful by means of computational abilities and provide an in-network processing of the sensory data. Moreover, the introduced hierarchy may affect the network stability due to the strict coupling between the cluster head and its members. For instance, a possible failure of a cluster-head can lead to the disconnection of its underlying area from the rest of the network. Moreover, in scenarios with mobile nodes it is improper to employ hierarchical network structures due to the great cost of maintaining such hierarchy.

2.2.2 WSN Protocol Stack and Application Designing

After the designing of the network topology, the developers continue with the modeling of the application and communication protocols that are employed by the WSN system. Typically, the protocols are organized based on a protocol stack specific for WSN systems, similar to the traditional Open Systems Interconnection (OSI) model, as it is presented in Figure 6. The WSN protocol stack provides power efficient communication and routing through the wireless medium, as well as it promotes further collaboration between the nodes of the network by utilizing data and networking protocols (Akyildiz et al., 2002). Moreover, It consists of the Physical Layer, the Data-Link Layer, the Network Layer, the Transport Layer and the Application Layer in addition to cross-layer services such as the Power, Mobility and Task Management services. However, the WSN stack formulation introduces iterations between the designing and the development phases so as to balance the major trade-offs, between flexibility, simplicity and efficiency. Typically, the developers first define the top services, and next continue to the lower layers. The cross-layer services are implemented last so as to readjust the main requirements if necessary.

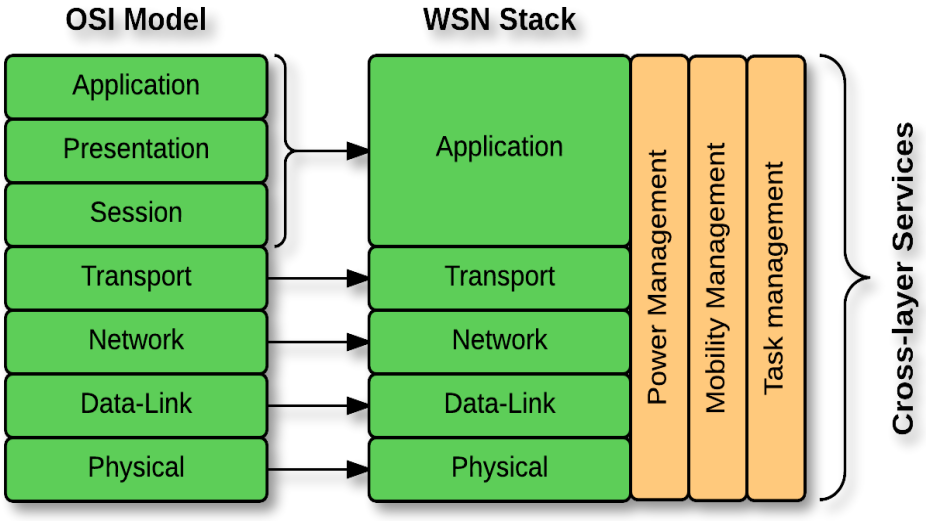

Figure 6. The WSN protocol stack

i. Physical Layer

The Physical Layer protocols specify the radio frequency and the carrier frequency generation. Also they are responsible for the modulation, encryption, and transmission of the signal as well as to detect neighbor and sensory signals. Considering the high consumption of energy in long-term communication over the wireless medium, the developers have to design and apply protocols that promote power efficiency. Moreover, due to the environmental radio noise, the physical layer protocols should ensure the transmission signal integrity.

ii. Data-Link Layer

The Data-Link protocols control the multiplexing of the data streams, the data frame detection, the medium access and the error control as well as they are responsible to provide reliable point-to-point or point-to-multipoint connections between the nodes of the network.

Moreover, the Medium Access Control (MAC) sub-layer addresses the issues of power conservation and data-centric routing. These unique features require a WSN specific MAC protocol that meets two goals. First, it should establish the links between the nodes, hop by hop, in order to be able to self-organize and transfer data. Second, it must provide fair sharing of the communication resources between the nodes. MAC protocols of traditional networking fail to accomplish these two goals, since power consumption is not a primary consideration in their design.

Another important functionality of the Data-Link layer is the error control of the transmission data, which can be classified into two groups, the Forward Error Correction (FEC) and the Automatic Repeat Request (ARQ). However, ARQ is not an optimum solution for multi-hop networks due to the additional overhead introduced by the retransmissions and thus affecting the overall network lifetime. On the other hand, FEC methods overcome this issue by enhancing correction algorithms within the node stack. However, developers have to take under consideration the high complexity of implementing such decoding procedures. Therefore they should design simple error control codes with low encoding and decoding complexity in order to provide a convenient solution for WSN systems.

iii. Network Layer

The Network layer specifies the optimal paths between the intermediate nodes so as the data packets can surely reach the central gateway, which can be a sink node or a base station. Moreover, every node executes a distributed algorithm in order to acquire and establish a common routing table. Recovery from system error, node failures or topology changes is essential in order to guarantee the availability of data dissemination paths (Koliousis & Sventek, 2007). The sensor nodes establish and maintain routes by employing either proactive or reactive data propagation technics.

The proactive protocols periodically monitor the links between the nodes in order to ensure connectivity and path availability amongst the active node. Therefore, the nodes advertise a possible variation of their routing state to the entire network so as to maintain a fully traversable and common network topology.

On the other hand, reactive protocols establish paths only upon request, for instance in response to a query, or an event. Generally, the node remains in an idle mode until it is required to generate a request packet or forward an incoming routing packet through its neighbor peers to the gateway. Next, the gateway responds over the reverse path with an Acknowledgment packet (ACK) so as to maintain an updated global routing table. Reactive data propagation is typically cheaper by means of overheads in the network traffic, since packets are generated only when it is necessary. Due to their simplicity, and inherent support for data on-demand, these protocols tend to be the best design choice for WSNs.

Moreover, routing protocols provide two basic mechanisms, which are the neighbor discovery, to discover and maintain connectivity with neighbor nodes, and flooding, to disseminate the network state to distant nodes.

iv. Transport Layer

The Transport layer is required in order to enable the users of the system to access the sensor field through the Internet or other external networks. However, the traditional Internet transport protocols such as User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) are limited due to the energy constrains and the data-centric communication imposed by WSNs (Iyer et al., 2005). For instance, it is well documented that the UDP protocol

does not provide any link reliability and TCP requires a global addressing scheme in order to transport data packets between the peers.

Hence, there is a necessity for designing transport protocols which could address the unique characteristics of WSNs, including the network topology, system services, data traffic parameters and resource limitations by means of energy constrains and medium access fairness. Moreover, the transport protocol should provide high energy efficiency, flexible reliability and optimum QoS, in terms of link throughput, delay and packet loss rate (Wang et al., 2006). Thus, the developers should design WSN transport protocols that enhance the functionalities of congestion control and loss recovery. These two components affect directly the overall network lifetime, reliability, and QoS parameters as it was previously explained.

v. Application Layer – Cross Layer Services

The key role of the Application layer is that it provides an abstraction of the underlying physical topology of the system, which further interacts with the actual WSN application. Moreover, this layer includes the appropriate interfaces that enable the users to monitor and manage the network infrastructure. In other words, the application layer includes the code of the main application as well as several management services.

However, due to the great diversity of possible WSN scenarios and purposes, the design of an appropriate protocol is a challenging task. Actually, the formulation of a generic layer scheme is almost impossible, considering the various requirements in different WSN applications. Thus middleware and cross-layer designs emerge in order to provide optimum solutions, by tightly integrating, either parts, or the total layered protocol stack. Such services benefit from the boundless implementation amongst the layers and promote efficiency by reducing the overall network overhead. For instance, MAC and routing protocols can be enhanced into one protocol so as to combine their functionality and minimize the end-to-end delivery latency, as proposed in (Du et al., 2007) and (Mouradian et al., 2014).

Additionally, further optimization of the network performance can be achieved by assigning different roles to the nodes, such as data storage, data aggregator or cluster head. These methods provide an in-network processing which can be activated either to individual nodes or to all the nodes of the network, according to the subjected scenario.

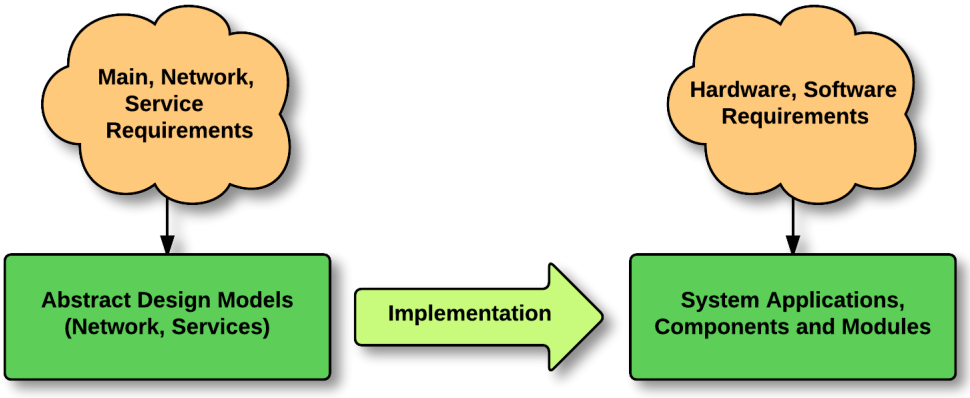

Figure 7. Implementation of Design Models to System Components

In cases where the network topology is stable the role assignment can be applied statically. However, self-organization is essential for dense and multi-hop networks and thus dynamic role assignment mechanisms emerge (Frank & Römer, 2005). Hence, these methods provide flexible solutions that further improve the system efficiency, as well as maximize the network lifetime. Nevertheless, the introduced complexity of the in-network functionalities requires greater development efforts and more detailed system models, so as to prevent resource mismanaging in terms of power consumption and traffic overhead.

2.2.3 Development Procedures

In the development phase of the project lifecycle, the researchers focus on implementing the designing abstractions of services and protocols into actual code that is processed by the node's OS (Figure 7). However, traditional operating systems do not address the unique characteristics of WSN applications. Therefore the research community has developed various lightweight operating systems specific for WSNs, which support main functionalities, such as dynamic component linking, clock synchronization, task scheduling, interruption management, memory allocation and networking support. Furthermore, the previously mentioned functionalities provide resource abstractions in a way to enable the developers of the system to employ high-level Application Programming Interfaces (APIs) independently to the underlying hardware (Dong et al., 2010).

The choice of the appropriate WSN OS is a critical decision since the supported programming models significantly induce the development

effectiveness of the application. Globally, there are two types of programming models, which are the event-driven and the multithreaded programming. It is most likely that application developers are more familiar with the traditional multithreaded model.

Nevertheless, this type of programming does not provide power awareness and thus being an unsuitable solution when it is applied to resource constraint systems such as WSNs. On the other hand, event-driven programming models tend to address the diverse nature and specifics introduced by WSNs but yet considered to be challenging for traditional application developers (Farooq & Kunz, 2011). Hence, researchers have implemented various hybrid WSN OS that adapt both programming models, such as the solutions proposed by (Dunkels et al., 2004), (Zhou et al., 2008), (Dong et al., 2011) and (Liu et al., 2014).

Moreover, additional efforts need to be taken by the developers, considering the implementation trades-offs between the code's efficiency, complexity and comprehensiveness. For instance, complex codding introduces difficulties in future updates, maintenance and reusability, even though it is strongly related to the requirements and the services of the subjected system. However, code clarity and comprehensiveness can be supported by following a component-based development approach in conjunction with coding primitives, such as variable naming and code commenting. On the other hand, by minimizing the introduced memory footprint and processing load of the code can contribute to resource awareness and improve the overall efficiency of the WSN application. Finally, as already mentioned in the previous Section, providing in-network functionalities by implementing cross-layer optimizations such as distributed coding, compression and encoding can increase the network lifetime and therefore its reliability.

2.3 Performance Evaluation – Validation

After the coding implementation of the system services and applications, the developers need robust validation tools and methods in order to verify the optimum functionality of the system components, prior to real deployment. Performance evaluation induces iterative procedures between the designing, development and testing phases in order to optimize the system modules and provide convenient solutions. After many years of technological evolution in the field of WSNs, has resulted in three main validation platforms, which are the sensor network simulators, emulators and the physical testbeds. However,

due to the complex nature of WSNs, each one of these methods has its own limitations when applied individually. Therefore, the researchers should combine all three in order to achieve more realistic conditions and retrieve valuable validation results (Coulson et al., 2012). A detailed presentation of the previously mentioned tools can be found in the upcoming Chapters; nevertheless a brief description is introduced as follows.

2.3.1 Simulation-Emulation

Simulation is the most widespread method for system validations across the research field of communication networks. Simulators enable scientists to evaluate their ideas and protocols in a convenient way by providing various levels of system abstractions in order to hide the complexity of low-level hardware functionionalities. Moreover, the researchers can investigate in depth the performance of the subjected solution by repeatably optimizing and isolating different system parametes in various scenarios (Papadopoulos et al., 2013). However, when it comes to the diverse nature of WSNs, there are many unique parameters that complicate the simulation fidelity, often making it unrealistic to test the instruction-level execution and its impact on the network lifetime. WSN simulations that do not consider the execution model of the node's OS, as well as the introduced synchronization issue of time drift due to hardware delays, produce inconvenient performance results (Riliskis & Osipov, 2015).

On the other hand, emulation is a special type of validation method that aims on bridging the gap between system simulation and real hardware performance. Moreover, the emulators try to provide the exact same instruction-set processing with real hardware by duplicating its functionality over detailed simulation models. Therefore, it is able to provide greater fidelity than simulation-based validations, as well as being more flexible and much cheaper than real hardware implementations such as purely physical testbeds. Despite its promising capabilities, emulation is a much less exploited approach in the field of WSNs. However, there are various scientific studies on the lower layers of the WSN systems, such as hardware drivers, networking and OS as well as cross-layer validation frameworks that incorporate emulation methods. For instance, emulators can compute the energy consumption of a particular WSN hardware platform according to detailed simulation of the radio model, as proposed in (Girod et al., 2004) and (Wu et al., 2007).

2.3.2 Testbeds

Due to the simulation limitations in the modeling realism of the deployment challenges, has resulted in an increased interest in developing real hardware laboratories known as testbeds. These platforms excel in the fidelity of the provided validations by enabling rigorous, transparent, and replicable testing of mature WSN designs and models (Papadopoulos et al., 2013). However, WSN testbeds are expensive platforms by means of, development, deployment, maintenance and overall cost. Additionally, most of them provide limited flexibility and heterogeneity due to the fixed network topologies, single type of supported mote platforms, protocol stack and OS. Therefore, the majority of scientists conduct experiments over custom, small-scale (i.e. typically up to tens of nodes) and local testbeds that are not open to the research community and do not promote reproducibility.

2.4 Real Deployment – Maintenance

Deployment of sensornets is considered to be the final phase where the nodes have to be set up in a geographic field so as to monitor the phenomenon of interest. However, deployment is a labor-intensive and complex task since environmental fluctuations often degrade the network performance or trigger system errors that could not be discovered during the evaluation phase (Ringwald & Romer, 2007). However, there are various issues that need to be addressed in order to result with an efficient deployment with minimum maintenance. Thus, we can categorize these issues based on three discrete phases related to deployment, which are the pre-deployment and deployment phases, post-deployment phase and final the re-deployment phase (Akyildiz & Vuran, 2010).

2.4.1 Pre-deployment and Deployment Phase

Initially, the nodes can be placed either one by one in a field or they may be deployed randomly according to the specific application. For instance, nodes can be deployed in an indoor laboratory or outside by dropping them from an airplane or by a missile, as well as they can be placed one by one either by a human or even a robot. However, in most large-scale scenarios the great number of involved nodes, in conjunction with the diverse environmental

conditions makes it almost impossible to place the nodes according to a carefully designed plan. Therefore the schemes for initial deployment must reduce the installation cost, increase the flexibility of the nodes topology arrangement and promote self-organization and fault tolerance functionalities.

2.4.2 Post-deployment Phase

After the deployment phase is completed, the network topology may change due to various reasons. Moreover, in scenarios where the nodes introduce mobility, their movement influences directly the topology of the network for-long time periods. On the other hand, the links between the nodes of the system can change in cases of radio jamming, environmental interference or noise while these factors affect the network topology for short-time periods. Also according to the sensing tasks of the nodes, the topology may change periodically, when certain nodes turn to sleep mode for a specific amount of time. Finally, the most critical case of topology change occurs when the nodes fail to participate in the network due to possible hardware deficiencies or lack of available power resources, which result in permanent changes. Consequently, the employed network protocols should enable the nodes to adapt to the aforementioned fluctuations in the topology.

2.4.3 Re-deployment Phase of Additional Nodes

The changes of the network topology introduced during the post-deployment phase, may require additional nodes to be deployed in order to overcome possible connectivity and fault tolerance issues. Therefore, additional nodes can be re-deployed at any time to replace the broken nodes or even to improve and expand the dynamics of the system. However the addition of new nodes introduces the requirement of network self-organization by utilizing special WSN and Ad-Hoc protocols.

3. Simulators

Due to the great complexity of deploying and maintaining large scale WSNs, troubleshooting procedures after real deployment are extremely expensive. Hence, the researches should provide reliable solutions that are evaluated over extensive testing. Testbeds of this scale are also expensive to be developed and need great effort to be managed (Mainwaring et al., 2002), (Ganesan et al., 2002). Therefore, WSN simulators are designed to provide a software platform that addresses the key aspects of the overall performance of a sensor network. Moreover, simulators enable handy designing, development, debugging and evaluation of new algorithms and protocols prior to hardware implementation.

However, it is almost impossible to duplicate the exact same conditions of real deployments. Therefore, studies based only on simulation practices affect the total credibility of the proposed research (Kotz et al., 2004). On the other hand, simulators can be considered essential for exploring WSN applications, acting as a common ground for the scientists to test their ideas. A single paper cannot puzzle out all the aspects of a complete application, but it can contribute in a way to promote further analysis and research in the field (Stojmenovic, 2008).

In order to effectively evaluate a study over simulations, it is important to have a good knowledge of the existing simulators and their capabilities. Up to the present day there are a variety of simulators that differ in their designing and modeling abilities. Thus, developers can choose the appropriate tool according to their needs and identify possible errors in their applications with ease such to provide further improvement. Nevertheless, lack of knowledge in the features of the available simulators can lead to inaccurate assumptions and buggy applications.

3.1 Simulator Design Requirements

Any WSN simulator should provide a diversity of underlying tools so as to replicate the behavior of real deployments. The basic requirements that should be implemented by a simulator in order to provide convenient results can be briefly described as follows.

i. Fidelity and Heterogeneity

In order to match any simulation needs, the software should provide accurate behavior of the underlying features of the simulated system. More specific, the models of the radio channel, the physical environment and network functionality should be faithfully implemented for optimum performance evaluation. Fidelity can be referred to both bit and temporal accuracies. For instance, the accuracy of simulating a transmission event is a correlation between the content of the transmission (bit accuracy) and the duration of the transmission time (temporal accuracy). Furthermore, by taking under consideration the nature of modern WSN applications, energy aware and heterogeneous mechanisms should be supported and replicated efficiently.

ii. Reusability and Availability

Globally, researchers are interested in challenging their novel proposals with keen current studies. Such feature enables detailed comparison between different simulated scenarios and applications so as to optimize new solutions. Hence, the simulators should provide a diversity of common WSN models that can be modified or even integrated with new ones. Those models should provide modularity in order to support fast prototyping by abstracting low-level functionality details. However, the existed variety of implemented WSN scenarios depends on the popularity and development support of the simulator in use. Recent protocols and applications that successfully involved and promoted evolution in the WSN field of study are mostly implemented in later versions. Also, availability can be referred to a cross-platform design of a simulator, by means of OS independency (Windows, Linux, etc).

iii. Performance and Scalability

Performance and scalability are critical tasks for modern WSN applications. As far as it concerns the simulators, those tasks depend on the underlying programming language that implements the simulation engine and its functionalities. It should not be confused with the programming language that implements the user-defined testing scenarios. Moreover, performance refers to the overall simulation speed, which is further defined as the ratio of the virtual simulated time to the physical run-time. Scalability is related to the ability of simulating large-scale networks and the effects of timely increasing network complexity on the overall performance. However, this mechanism is

limited to the memory, processing and storage features of the computer hosting the simulator.

iv. Rich-Semantics Scripting Languages and Graphical User Interface

Due to the great amount of information involved in the simulation process, rich-semantics scripting languages should be supported, in order to define simulation settings and retrieve output results. The Graphical User Interface (GUI) is considered to be a major requirement because it enables convenient debugging, fast network modeling and handy visualization of the results without the need to employ third party software. Inexperienced users can get an easier control of the simulations by using a simulator that provides a user friendly GUI.

3.2 Discrete Time Simulations

WSN simulators are discrete time software platforms, which means that simulations are driven by events that occur over discrete time. Furthermore, the architecture of the core mechanism comprises the scheduler and the simulation models. The simulation models implements the functionality and the logics that model a physical condition. On the other hand, the scheduler is responsible to provide the appropriate collaboration between the models, by managing the simulation time in accordance to the simulation models so as to induce and fluctuate the simulated system at discrete time points. The changes on the simulated system are introduced by varying the values of some state variables during simulation time.

The scheduler can be designed as *Time-Driven* or *Event-Driven*, based on the way that the time points are divided, thus promoting the simulation events. Every time point introduced by the scheduler, which can be either a step or an event, is further processed by a function that is typically called handler. The handler is responsible to invoke the proper simulation models that need to cooperate in order to update the current state of the simulated system variables, if necessary.

```
1       while(true)
2       {
3               //Event queue
4               if(!empty(eventQueue))
5               {
6                       // get the first event from the queue
7                       Event e = dequeue(eventQueue);
8                       // progress simulation time according to the event
9                       timeProgression(e.time);
10                      // call the handler of the specific event
11                       e.handler.handleEvent();
12              }
13      }
```

Code Listing 1. Example of an Event-Driven Simulation

i. Event-Driven

The event-driven design induces the scheduler to divide the time into points that correspond to simulation events, for instance sending or receiving a packet, some change in the sensory field of interest etc. Differently to time-driven designs, in event-driven schedulers there are various handler functions corresponding to every event. This method is considered to be more accurate than time-driven scheduling due to the ability of evaluating the simulated system only when the events occur, as presented in Code Listing 1. However, this method of scheduling introduces complexity and requires strong computational capacity and development effort to conduct event-driven simulations.

ii. Time-Driven

When the scheduler follows a time-driven design, the simulation time is divided into further points that stand off equal time length, for instance one point every minute, hour etc., always according to the tested scenario. Consequently, the events occur based on a list containing respectively timestamps. Globally in a time-driven scheduler there is a single handler function that performs evaluations of the system on every time step, regardless the presence of changes in the simulated system.

```
1       for (every t)
2       {
3               //progress simulation time by t
4               timeProgression(currentTime+t);
5               //call simulation handler
6               simulationHandler();
7       }
```

Code Listing 2. Example of a Time-Driven Simulation

Moreover, a time-driven design is considered to be more suitable in cases where the changes of the simulated system happens more or less periodically on some predictable time points, as presented in Code Listing 2. Hence, without the presence of overheads in managing the events as in event-based scheduling, simulations that follow a time-driven design are more convenient to implement.

3.3 Simulation Models

The key aspect of simulators is to provide convenient simplifications of real conditions and interactions between the underlying elements of a sensor network and the physical environment. The structure of the simulated condition is implemented in several models that individually define a specific feature or operation, along with their relations (Figure 8). Moreover, the simplifications should be well defined and detailed in order to acquire valuable results.

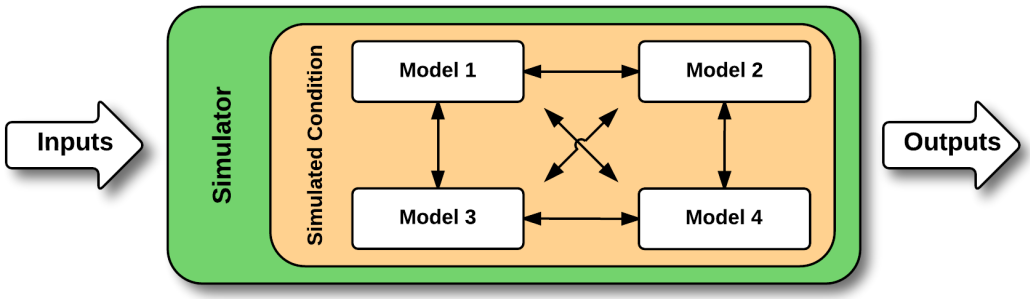

Figure 8. Typical Simulator Structure

Hence the simulated models should be based on realistic assumptions. However, the complex of precise modeling capabilities, in accordance with large-scale scenarios, increases dramatically the utilization of resources and finally affecting the overall simulation performance. Therefore, the fundamental trade-off is to provide accurate modeling while meeting the aforementioned simulator design requirements. In addition, many methods have been used to deal with scalability issues such as component-based design and parallel simulation. In a parallel simulation, the simulated components are distributed over several Central Processing Units (CPUs), where the sub-programs are concurrently executed, while the simulator scheduler is responsible for the overall synchronization. Despite the complexity of real systems, a general description of the main component models is introduced as follows.

3.3.1 Network Model

The network model is a broad simplification of the system network, which further incorporates the following individual models. Figure 9 depicts these models and their relations.

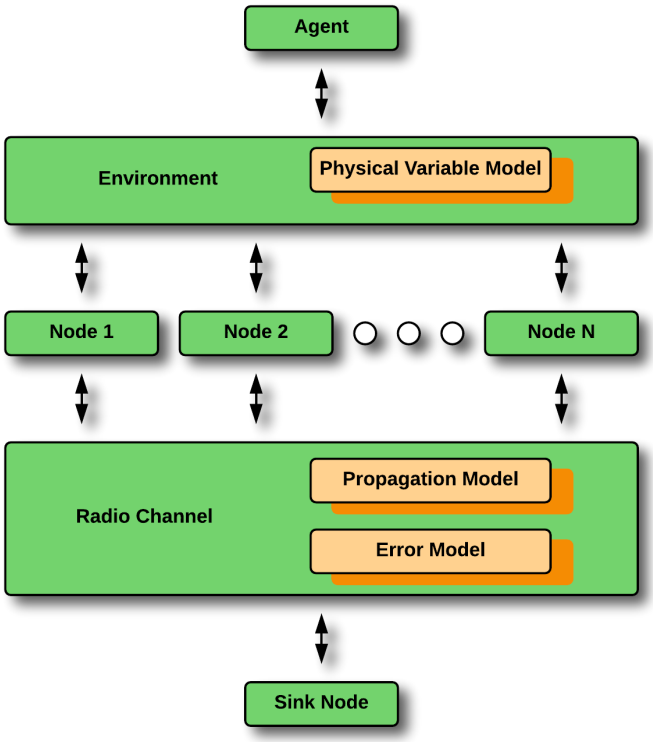

Figure 9 Typical WSN Network Model

i. Nodes

The node is the physical device that monitors a set of physical variables that simulate a physical condition of interest. Also the nodes are connected on a common radio channel in order to communicate with each other. The protocol stack that is implemented as part of the inner node model defines the connection and communication mechanisms. Unlike the classical network models, sensor networks introduce complex concepts such as mobility, energy efficiency, and sensory capabilities that are constrained by the physical environment. These concepts are also implemented as part of the inner node structure that further interacts with the environmental model. A typical simulation scenario can involve several numbers of nodes, however large-scale networks that introduce great scalability are still a challenging task. More detailed description about the node functionality is given in the Node Model section below.

ii. Environment

The environmental model is the main component that differs WSN models from the rest network types. This feature simulates real environmental conditions with sensed physical variables. Moreover, its basic functionality is to generate and propagate events that further trigger the node to initiate an activity of interest such as communicating with other nodes. Physical variables of interest include data such as temperature, seismic waves, sound, water pollution etc. However there are some simulators that they employ an agent for each physical variable. In other words, it means that the events generation is separated from the environmental model.

iii. Radio Channel

The radio channel consists of the propagation and the error models. The propagation model specifies the diffusion of the radio signal among the nodes of the network. The environmental model varies due to insertion loss and its effect on the signal quality. Furthermore, there are several propagation models that differ in complexity, however more complex models are globally more resource demanding, which can further affect the overall performance trade-off. In more detailed models, the use of a terrain component is connected to the environment and radio channel models and it is taken into consideration so as to compute the propagation, by influencing several

physical magnitudes. The error model addresses the random phenomenon that affects the rates of packet reception as well as packet loss. Normal distribution, Markov chain or empirical models are typically used and optimized in order to implement such error models.

iv. Sink Nodes

The sink nodes are special nodes that receive and process data from the rest sensor network. They are used to provide fast advancement of valuable information that is related to sensory data collected by the network nodes. The use of sink nodes depends on the subjected application and scenarios that are tested through the simulator.

v. Agents

The agent acts as the generator of events of interest in order to trigger the nodes. More specific, the agent may cause a variation in a physical magnitude, which is propagated through the environment and impels the sensors of the nodes. This component typically is implemented as part of the protocol tier that is described below, however it can be more effective if implemented separately from the node and the environment models (Figure 9).

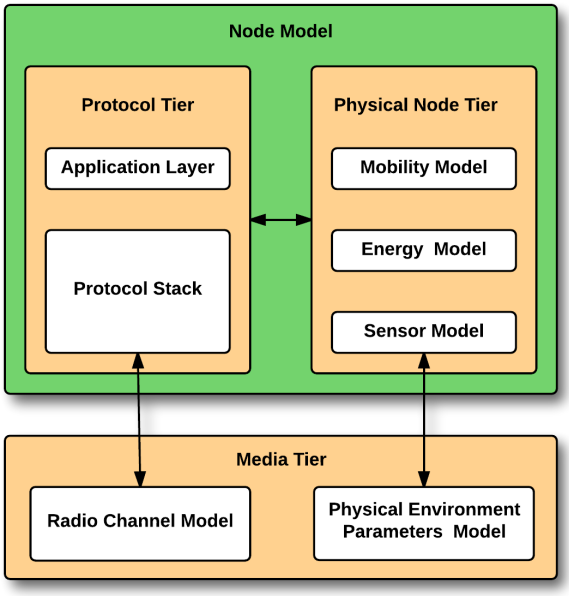

Figure 10. Tier-based Node Model

3.3.2 Node Model

The overall node mechanism depends on various models that incorporate in such way so as to form a cross-layer functionality. However, in order to facilitate the development process, these models are grouped into further classes according to their purpose. Those abstract tiers and their comprised models are briefly described below and are illustrated in Figure 10.

i. Protocol Tier

The protocol tier groups all the appropriate communication protocols. Globally, the protocol tier contains three layers, which are a MAC, a routing and a specific application layer. The functionality of the protocols depends on the state of the physical tier that is presented below, for instance a routing protocol may consider energy constraints in order to decide the packet route. Hence, efficient methods that enable information interchange between the tiers must be implemented.

ii. Physical Node Tier

The physical node tier simulates the resources of the node, by means of hardware and its effects on the performance, lifetime, capabilities and functions of the node. Actual composition of this tier may change depending on the specific application. Typically, that tier consists of the physical sensor model, the energy model and the mobility model. Physical sensors describe the sensory behavior of the monitoring hardware. A critical feature of WSN applications is the energy model, which simulates the power consumption during common node activity such as sensing, data processing and communication. Also there is a mobility model that defines possible movement of nodes by changing their position parameters during simulation time. Generally, the mobility is implemented either by a movement vector in accordance to the initial position or by a list of positions linked to timestamps.

iii. Media Tier

The media tier acts as a common ground between the nodes and the simulated physical environment. A node can interact with the environment through an ordinary radio channel that is further affected by the physical parameters as described previously in the radio channel Section.

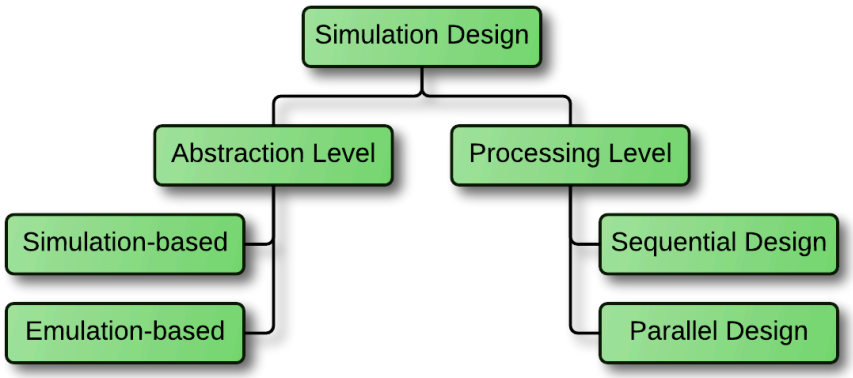

Figure 11. Classification of Simulation Designs

3.4 Simulation Design

As described in previous Section, simulation models implement the logics and functions of simulated WSN applications. Furthermore, the simulation models are developed based on various designs that also play important role in order to implement the design requirements of WSN simulators. A broad classification of possible simulation designs is applied based on two key aspects that are the abstraction and multiprocessing capabilities, as illustrated in Figure 11.

3.4.1 Abstraction Level Design

The simulators can follow two types of designs based on their abstraction capabilities in order to address the requirements of the simulated system.

i. Simulation Based Design

Globally, simulations can be used to test the performance of new applications without considering constrains that may introduced by the employed hardware. Moreover, simulators provide handy modeling of the interior features of the subjected physical systems by simplifying the software development process for a particular WSN application. Simulation based designs do not provide fully accurate evaluation. However it allows fast prototyping because it provides handy development of WSN systems in high-level abstractions.

ii. Emulation Based Design

The emulators are a special type of simulators that aim on increasing the overall evaluation fidelity by modeling the subjected system in a realistic manner. In other words, emulators require cross-layer implementations of the models in addition to low-level functionalities. Furthermore, emulation based designs can improve the simulation performance, by employing real sensor motes and implementing their corresponding instruction set, so as to support native execution of actual WSN code. Emulators provide accurate simulations of WSN applications because an emulator executes the same machine code that runs on a real sensor node processor. Consequently, this type of design introduces overheads due to complex models and requires greater resources than a typical simulator.

3.4.2 Processing Level Design

Simulators can be divided in two types based on the way that they process the simulation events.

i. Sequential simulators

Simulations that follow a sequential design introduce the simulated events in a queue that is further addressed by a single processor.

ii. Parallel simulators

Simulators that employ a parallel design introduce the simulated models in a distributed system that further addresses individual processes either to a multi-core processor or to multiple processors. Therefore, this type of design can improve the overall simulation speed and performance.

However, simulations based on distributed designs are able to provide limited speed and scalability due to great complexity in providing the temporal relations of the interactions between the simulated models. For instance, in a scenario that sensor nodes are simulated in parallel, their simulation speeds may vary due to differences in either the number of inputs or node models, in addition to lack of available processors so as to simulate all the nodes simultaneously. Since nodes may get simulated at different speeds, it becomes critical to preserve the causality of events for optimum simulation-based evaluations (Titzer et al., 2005), (Jin & Gupta, 2008).

In order to overcome the above synchronization issues many protocols have been proposed. Globally, these parallel synchronization protocols can be classified into two groups according to their approaches, which are the conservative and optimistic protocols.

In a conservative approach, a process is able to progress in simulation only when the causality is preserved. However, it is possible to result in a deadlock. Therefore conservative protocols require additional messages to be enhanced in order to transmit the local time between the different simulated processes, thus introducing a major overhead. On the other hand, an optimistic protocol allows the simulated process to progress in simulation time until it discovers a violation of casualty, where the process has to move backwards in time along with cancelling all the transmitted messages. This rollback is achieved by periodically saving the simulation state so they can be handled as checkpoints. Therefore, the major overhead in an optimistic approach is the processing of the rollback actions (Lim et al., 1998).

3.5 Taxonomy of Simulators

Simulators are the most wide spread evaluation tool amongst the WSN research community for designing new applications. The key features that play crucial role in the simulation process are the models of the subjected nodes, network and environment as described previously. However, environmental modeling is still a challenging task since the majority of the existing tools provide poor designs by means of details and abstraction (Hammoudeh et al., 2008). Additionally, many available proposals of WSN simulation frameworks, like the proposals of (Guestrin et al., 2004), (Chiasserini & Garetto, 2004) and (Gracanin et al., 2004), do not provide detailed environmental models. On the other hand, modern research studies try to address this controversy and provide more specific information and detailed frameworks so as to increase simulation efficiency, as the proposals of (Merrett et al., 2009), (Lo et al., 2007), (Corke et al., 2010) and (Ferencik et al., 2010).

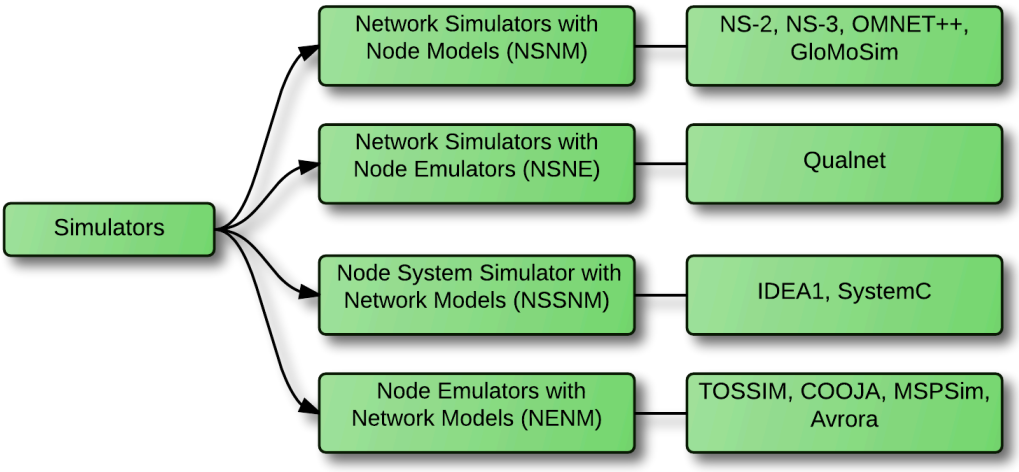

Figure 12. Simulators Taxonomy (Du et al., 2014)

Therefore, the following taxonomy will focus only on the simulators' node and network modeling capabilities in accordance to the employed abstraction level design. Hence, WSN simulation tools can be divided into four categories, which are the Network Simulators with Node Models (NSNM), the Network Simulators with Node Emulators (NSNE), the Node System Simulator with Network Models (NSSNM) and finally the Node Emulators with Network Models (NENM), as depicted in Figure 12. A brief description of each category can be found below.

i. Network Simulators with Node Models (NSNM)

Network simulators with node models, typically, implement event-based scheduling designs among the underlying simulation models, such as the radio channel, the node and the network models. However, the main objective of this type of simulators is to emphasize on network modeling capabilities, which is the predominate entity. Hence, the node models are implemented in higher abstraction level, by means of functional complexity. Many popular simulators belong to this group, for instance NS-2, NS-3, OMNET++ and GloMoSim.

ii. Network Simulators with Node Emulators (NSNE)

Network simulators with node emulators, aim on addressing the advantages of the overall abstraction level classification, by employing network

simulations in conjunction with node emulations. Furthermore, on the network simulation part, the developer implements the details of the network models. On the other hand, the node emulators process natively the nodes' instruction set, which provides accurate performance results. However, the communication between the network simulator and the node emulator introduces overheads that lead to time-consuming evaluations. A popular example of simulator that belongs to this group is Qualnet.

iii. Node System Simulators with Network Models (NSSNM)

Node system simulators with network models, globally utilize System-Level Description Languages (SLDL), such as SystemC, in order to model the underlying node system. Moreover, SLDL enables researchers to design simultaneously high-level abstractions of the hardware and software components, as they would have been in a real system. Hence, researchers can focus on the overall functionality of the system instead of its implementation details, which promotes convenient evaluations of different architecture alternatives. Finally, the NSSNM compared to NSNM can provide faster execution due to the SLDL effectiveness (Du et al., 2011). Some examples of this type of simulators are, IDEA1 (Du et al., 2011), SystemC/MSPSim platform (Stecklina et al., 2011).

iv. Node Emulators with Network Models (NENM)

A node emulator with network models can be considered as a conjunction between two further sets, which are the Instruction Set Simulator (ISS) and a WSN OS emulator. The ISS is used to simulate specific microcontrollers and processors, and eventually collaborates with an OS emulator that is used to emulate the execution of node application code over embedded WSN OS such as TinyOS, and Contiki. This type of simulators provides high timing accuracy of software execution compared to real implementations due to the ability of processing the embedded software directly in the simulation framework without modifications (Eriksson et al., 2009). Some examples of simulators that belong to this group are TOSSIM, COOJA, MSPSim and Avrora.

Each one of the aforementioned simulator types aim on fulfilling different simulation requirements. Those identifying features that separate each type are briefly presented in Table 1 as proposed in (Du et al., 2014).

Table 1. Characteristics of the different simulator types

Simulator Type	Advantages	Disadvantages
Network Simulators with Node Models (NSNM)	• Network modeling • Radio channel modeling • Scalability	• Simple power model • Simple timing model
Network Simulators with Node Emulators (NSNE)	• Network modeling • Detailed channel modeling • Detailed timing model	• Scalability
Node System Simulators with Network Models (NSSNM)	• Network modeling • Radio channel modeling • Scalability	• Moderate timing accuracy • Moderate power accuracy
Node Emulators with Network Models (NENM)	• High timing accuracy • High energy accuracy	• Simple network modeling • Simple channel modeling • Scalability

3.6 Survey of WSN Simulators

As already mentioned in the previous Section, according to the simulator taxonomy there are a number of available tools that can be employed based on the requirements of the subjected research. Moreover, by taking into consideration the popularity of the simulation tools presented in Chapter 5, a brief description of these simulators is presented below. Each further description is introduced with the following format: a summary of the simulator, the programming language in use, its key features and finally its limitations, as summarized in Table 2.

Table 2. Summarized simulators characteristics

Simulator	Language	Key Features	Limitations
NS-2	• C++ • Tcl	• Modular design • Extensible • Diversity of predefined models • Visualization tool called NAM • Packet level execution	• Long learning curve • Predefined models can not be modified • Standard application layer models
NS-3	• C++ • Perl • Python	• Open-source, build from scratch • Realistic simulation models • Can be employed as real-time emulator • Pcap output files for further visualization by third party tools • Supports both IP and non-IP based networks	• Not backward compatible with NS-2 • Lack of credibility • Scalability depends on the host PC's computational and memory resources
OMNET++	• C++ • NED	• Open-source • Modular design • Extensible • GUI based on Eclipse IDE • Extensions for real-time simulation, emulation and SystemC models • High scalability	• Lack of accuracy • Few predefined models • Difficult to combine the predefined models since they have been developed by different research teams

Simulator	Language	Key Features	Limitations
GloMoSim	• PARSEC • C	• Sequential and parallel simulation environment • Modular design • Extensible • Suitable for simulating mobile wireless IP networks • High scalability • User-friendly GUI	• Lacks the ability of providing accurate simulations • Discontinued since 2000
Qualnet	• PARSEC • C • C++	• Based on GloMoSim • High fidelity • Modular design • Sequential and parallel simulation environment • Individual measurements on each layer • Java-based GUI	• Commercialized software • High CPU utilization • The GUI is slow in most computers
TOSSIM	• nesC • C • C++ • Python	• Open-source • Simulation of TinyOS-based applications • High scalability • Powerful GUI	• Lack of power consumption models • Limited to TinyOS applications • Lack of heterogeneity
COOJA	• Java • C	• Open-source • Extensible • Simulation of ContikiOS-based applications • Simulation of heterogeneous networks • Convenient transition to real deployments • Binding with MSPSim • GUI and visualizer • JNI supports third-party debugging tools	• Limited to the Javas' heap memory • Slow simulation time

Simulator	Language	Key Features	Limitations
MSPSim	• Java	• Open-source • Instruction-set emulation of MSP430 microcontroller • Realistic simulation • Accurate timing • High scalability • Modular design • GUI • Convenient transition to real deployments	• Limited to the Javas' heap memory
Avrora	• Java	• Open-source • Instruction level simulator • Cycle accurate • Language and OS independency • Based on Atmel AVR microcontroller • High scalability	• Does not model clock drift • Lack of GUI • Does not support mobile scenarios • 50% slower than TOSSIM
Matlab	• C • Fortran, • Python • C++ • Perl • Java • ActiveX • .NET	• High-scripting language • Flexible, reliable • Hundreds of build-in mathematical functions • Simulink and other frameworks for WSN simulation • Extendible • Friendly GUI	• Commercial software • Interpreted language • Slower performance
EnergyPlus	• Fortran • C++	• Energy-management simulation tool • Modular design • Interface for external programs	• Lack of GUI • Simulation based on input files

3.6.1 Network Simulator 2 (NS-2)

i. Summary

Network Simulator 2 is a discrete event simulator that aims on networking research in general (The Network Simulator - NS-2, 2011). Network simulator was developed in 1989 as an alternation of the REAL simulator (Keshav, 1988), and since then three major versions has been released NS-1, NS-2 in 1996 and NS-3 in 2008. As far as it concerns NS-2, it is designed based on a modular approach that enables effective extensibility (Sarkar & Halim, 2011). Furthermore, this simulator provides a variety of predefined models in order to support simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. However, the upgrade of simulating wireless network technologies, such as LTE, MANETs and WSNs, was introduced in later versions. Moreover, the simulation scenarios that are tested over the NS-2, follows strictly the OSI reference model.

ii. Simulation language

NS-2 simulations are conducted based on a combination of C++ functions, which models the behavior of the simulation nodes, and Tcl scripts that control the simulation process and specify further features, such as the network topology (Chaudhary et al., 2012).

iii. Key features

NS-2 is a popular simulator for WSN application due to the provided extensibility in conjunction with a great number of predefined protocols and models that are available within the simulator package. It follows an object-oriented design that allows straightforward implementation of new protocols. The key features of WSN systems that are supported include sensor channel models, power models, lightweight protocol stack, hybrid simulations and finally, scenario generation tools. Moreover, a visualization software tool called Network AniMator (NAM) is employed in order to support topology layout, packet level animation, and various data inspection mechanisms (Nam: Network Animator, 2002). The simulations are executed in packet level, thus producing detailed results and enabling handy debugging.

iv. Limitations

NS-2 requires a long learning curve and advanced skills in order to conduct valuable and repeatable simulations. A major drawback of NS-2 is that a user is unable to modify the provided protocols and models (Chaudhary et al., 2012). Furthermore, the packet formats, the energy models, the MAC protocols and the hardware models differ between various WSN systems. A unique characteristic of sensor networks is the fact that the application layer interacts often with the lower protocol stack, nevertheless NS-2 lacks of the ability to provide a modifiable application model.

3.6.2 Network Simulator 3 (NS-3)

i. Summary

The NS-3 simulator (NS-3, 2011) is a free open-source discrete-event network simulator especially devoted for research and educational use. The NS-3 project started in 2006 and it was first release to the public on July 2008 with the version 3.1. Up to the current day the latest version is 3.23 that was released on May 2015. NS-3 was developed from scratch and it is not an extension of NS-2, thus not backward compatible (Chaudhary, Sethi, & Keshari, 2012). However, NS-2 community still continues to provide support and maintain the simulator package in order to study transition and integration mechanisms to NS-3. Furthermore, new features have been included compared to NS-2, in order to provide detailed simulation tests of any networking technology. Every three months, new stable version of NS-3 is shipped, containing new developed models that are documented, validated, and maintained by researchers.

ii. Simulation language

NS-3 simulations are conducted entirely in C++ with optional use of Python bindings. The network component models as well as the user-defined simulation scenarios are implemented either in C++ or PERL, however differently to NS-2, they can be totally written in C++ programming language.

iii. Key features

NS-3 is open-source, and the project team maintain an open environment so as researchers across the globe to be able to contribute and share their proposals. Furthermore, the ns-3 software infrastructure encourages the development of simulation models, which are sufficiently realistic to allow NS-3 to be used as a real-time network emulator, interconnected with real systems. For instance, users can send and receive NS-3 generated packets on real network devices, while NS-3 serves as an interconnection framework in order to provide a link between the virtual machines. Thus, many existing real world protocol implementations can be reused within NS-3. Furthermore, NS-3 generates Pcap (Packet Capture) packet trace files that can be processed by third-party visualization and tracing software, such as Wireshark (Wireshark, 2006), NetAnim (Riley, 2012) and Gnuplot (Gnuplot, 1986). Finally, its simulation core supports research on both IP and non-IP based networks. (NS-3, 2011)

iv. Limitations

One limitation of NS-3 is the credibility of the simulation results. The employed network simulation models are modifications of already available ones, and possible malfunctions may transfer and affect the performance of the simulated systems. Also the scalability of the introduced simulation scenarios is constrained to the available memory capacity and computational capability of the computer hosting the simulator.

3.6.3 OMNET++

i. Summary

OMNET++ is an extensible, modular, component-based and discrete-event simulation framework, primarily for building network simulators for wired and wireless networks. However, support for specific domain networks, such as sensor networks, wireless ad-hoc networks, MANETs etc., is provided by model frameworks, developed as individual projects like Castalia (Castalia, 2007). OMNET++ offers an Eclipse-based IDE, a graphical runtime environment, and a powerful GUI library for animation, tracing and debugging support. There are extensions for real-time simulations, network emulations, database integrations, SystemC system models, and several other functions.

Getting started with it is quite simple, due to its clean design. Finally, OMNET++ was developed to fill the gap between open-source and research-oriented simulation software tools such as NS-2 and the expensive commercial alternatives like OPNET.

ii. Simulation language

The OMNET++ framework is totally implemented in C++ programming language. However the underlying network models can be grouped in broader structures called components by using NED, which is the topology description language employed by OMNET++ (NED, 1998).

iii. Key features

OMNET++ is an open architecture simulation environment with an embeddable simulation kernel that enables extensibility and handy integration of new protocols and network technologies. The ease of modifying the sensor network properties and its scalability makes OMNeT++ an excellent tool for simulation-based evaluations of WSN applications and systems. Moreover, the provided graphical user interface enables handy tracing and debugging procedures.

iv. Limitations

A drawback of OMNET++ is that it lacks of available protocols in its library, compared to other simulators like NS-2. However, OMNET++ is becoming a popular tool and new contributions have extended the initial framework. Nevertheless, most of the available models have been developed by different research groups and do not share a common architecture, thus combining them is a challenging task. Another major concern is the accuracy of the simulated models. The authors in (Colesanti et al., 2007) prooved that simulation performance results retrieved from OMNET++, differ significantly from real experimental results.

3.6.4 GloMoSim

i. Summary

Global Mobile Information System Simulator, known as GloMoSim (Zeng et al., 1998) is a discrete-event sequential and parallel simulation environment for wireless networks. Its library is classified according to the OSI reference model and consists of modules that respectively model a specific procedure in the protocol stack. GloMoSim follows a modular design in order to support extensibility, thus enabling researchers to modify, develop and share new modules and protocols. Moreover, simulation scenarios can be executed in shared memory and distributed computers by employing a variety of synchronization protocols in order to improve simulation performance. Moreover, several choices are provided within its library for radio propagation, Carrier Sense Multiple Access (CSMA) MAC protocols and implementations of UDP and TCP. GloMoSim is suitable for simulating mobile wireless IP networks. However, the development and support of the GloMoSim project has been discontinued since 2000 and replaced with the commercial Qualnet project.

ii. Simulation language

Simulations in GloMoSim are conducted using PARSEC (Bagrodia et al., 1998). PARSEC is a simulation language implemented in C, by the Parallel Computing Laboratory at UCLA for sequential and parallel execution of discrete-event simulation models, thus enhancing this ability to GloMoSim.

iii. Key features

GloMoSim is a powerful simulation tool due to its ability to execute parallel models and scenarios in a distributed manner. This can be achieved by employing one of the three different conservative synchronization algorithms that are provided within the package of the simulator. The provided algorithms are the null message protocol (Misra, 1986), the conditional event protocol (Chandy & Sherman, 1989) and the Accelerated Null Message Protocol (ANP) (Jha & Bagrodia, 1993). The choice of the conservative runtime algorithm is introduced as an option in the execution command. Hence, GloMoSim is able to handle large-scale scenarios with optimum performance. Furthermore, like most simulators, GloMoSim is an extensible simulator that

implements the underlying network models as modified modules, thus enabling handy development of new proposals.

iv. Limitations

GloMoSim provides basic protocols and functionalities for wireless communication networks. However, researchers like in (Alageswaran et al., 2013) try to address this deficiency by implementing and evaluating WSN specific protocols though GloMoSims' simulation engine. Moreover, GloMoSim similarly to NS-2, lacks the ability of providing accurate simulations of the packet formats, the energy models, and the MAC protocols functionalities compared to real WSN systems. Finally, GloMoSim project has been discontinued since 2000 and commercialized as Qualnet project.

3.6.5 Qualnet

i. Summary

Qualnet is a commercial communication simulation platform based on the core of GloMoSim. It was released on 2000 by Scalable Network Technology (SNT) for commercial use in order to simulate the functionality of real wired and wireless communications networks, as long as the underlying network devices (Qualnet, 2008). Qualnet extends significantly the availability of build-in protocols and models compared to GloMoSim. Furthermore, it supports advanced wireless modules and provides powerful tools for handy designing, developing and debugging of new ideas. Qualnet, similarly to its predecessor is a discrete-event sequential and parallel simulator.

ii. Simulation language

Qualnet uses PARSEC to conduct simulations, therefore the models and the build-in libraries are implemented in C and C++ programing languages.

iii. Key features

Qualnet aims on providing high fidelity by supporting many popular protocols and network device models. Moreover, due to its parallelization design it can produce the same fidelity for different scale scenarios with increasing

scalability. Qualnet has a modular layer structure that enables comparative performance evaluations of alternative protocols by collecting individual measurements on each layer.

iv. Limitations

Compared to most available simulators, Qualnet is a very powerful simulation tool that provides convenient simulation results, however it is not an open source project. Users have to pay in order to acquire a licensed Qulanet package. Nevertheless, the worst drawback of Qualnet is the extreme high CPU utilization of its Java-based GUI, which runs very slow on most machines.

3.6.6 TOSSIM

i. Summary

TOSSIM is a discrete event simulator and part of the TinyOS project (Levis, 2006), which is an embedded operating OS specialized for wireless sensor networks, both developed at Berkeley University of California (Levis et al., 2003). Thus, TinyOS applications can be compiled directly into the TOSSIM framework, which can further simulate thousands of nodes running complete applications. Furthermore, TOSSIM replaces the low-level components of a TinyOS system, such as the Analog-to-Digital Converter (ADC), the Master Clock, the EEPROM and several of the components in the radio stack in order to emulate their real behavior. Programs developed through the TOSSIM framework can be transferred to real motes without any modification, thus enabling researchers to easily transition between running an application on motes and in simulation. Also it provides a GUI that enables handy visualization, designing and debugging of running simulation scenarios in a controlled and repeatable environment. Moreover, TOSSIM simulates the network functionality at bit granularity due to low-level assumptions. Therefore, TOSSIM aims on simulating the execution of TinyOS applications, rather than simulating the real world. While TOSSIM can be used to understand the behavior observed in the real applications, it does not take into consideration all the details, and should not be used individually for absolute evaluations.

ii. Simulation language

TOSSIM framework was developed in nesC (Gay et al., 2003), which is an extended library of C programming language, aiming to provide a component-based programming model for embedded systems. Furthermore it supports Python and C++ programming languages.

iii. Key features

TOSSIM is an open source project with a large community that provides online documentation and support. Moreover, it comes with a visualization tool named TinyViz, which enables users to design easily WSN applications and monitor their functionality. Simulation scenarios are modeled in low-level abstractions that enable detailed debugging and support for large-scale experimentation including thousands of nodes. Overall, it is very simple and powerful WSN emulator for TinyOS-based networks.

iv. Limitations

TOSSIM is limited to TinyOS implementations and is not able to simulate any other type of network or protocol. Moreover, power consumption models are not supported, however users can employ PowerTOSSIM (Shnayder et al., 2004), which is an extenstion of the TOSSIM framework in order to simulate accurately the power consuption of the nodes. Furthermore, heterogenous networks are not supported because all the simulated nodes share the same TinyOS application.

3.6.7 COOJA

i. Summary

COOJA is an open source and flexible simulator for the Contiki OS specialized for sensor node (Dunkels et al., 2004), which allows cross-layer simulations between the different levels of the WSN system, such as the OS, the network and the instruction set levels. Simulation scenarios provide low-level abstractions of the underlying mote hardware in conjunction with high-level abstractions of the network behavior. It was developed following an extensible design through all of the simulated models, including the sensor node platform, the radio transceivers and propagation models etc. Moreover,

COOJA is able to simulate heterogeneous networks with different types of nodes, by means of hardware and software. The node type may be shared between several nodes and determines properties common to all these nodes. Contiki applications can be executed either as native code on the host CPU or by employing a specific instruction set emulator for MSP430 boards named MSPSim (Eriksson et al., 2009). Additionally, the applications developed through COOJA can be transferred directly to real mote hardware, thus minimizing the transition effort to real deployments. Finally, COOJA enables users to save simulation state in order to later restore the simulated scenarios or even skipping back simulation over time (Österlind et al., 2006).

ii. Simulation language

COOJA simulation engine is implemented in Java programming language, and all the interactions with C-based Contiki code are addressed through the Java Native Interface (JNI).

iii. Key features

COOJA is a free, open source, code level simulator for sensor networks that simulates nodes hosting Contiki OS, thus enabling convenient transition to real deployments. However nodes with different OS and characteristics may be included in a simulated scenario. Moreover, Java-based nodes provide fast simulations but do not run deployable code. On the other hand, by emulating nodes with MSPSim provides more detailed results compared to Java-based nodes or nodes running native Contiki applications. Nevertheless, native code simulations are more efficient than node emulations and additionally they evaluate deployable code. COOJA provides extendibility in two ways. First, by modeling the hardware peripherals of the simulated nodes as interfaces, which enable the Java simulator to detect and trigger events such as incoming radio traffic. Secondly, all the interactions between the simulator engine and the simulated nodes are performed via plugins, for instance starting or pausing the simulation progress. Moreover, a GUI is provided in order to design and develop the simulated WSN system, with an additional visualization tool named TimeLine, for presenting the radio traffic and radio usage of the simulated network (Österlind et al., 2010). Finally, the Java Native Interface enables Contiki code debugging by employing third party tools like GDB (GDB: The GNU Project Debugger, 2006).

iv. Limitations

COOJA, due to its extendibility, is limited to the Javas' heap memory. Simulating many nodes with several interfaces requires a lot of calculations and thus increasing the simulation run time.

3.6.8 MSPSim

i. Summary

MSPSim is an open source instruction set emulator that is able to simulate complete WSN motes such as Tmote Sky, as well as custom WSN motes based on Texas Instruments MSP430 microcontroller. MSPSim aims on providing realistic simulations with accurate timing in conjunction with handy debugging control. Furthermore, extendibility is supported through a variety of available build-in implementations of different peripheral devices as components, which are further simulated based on a discrete-event approach. Moreover, the ability to process and interpret real hardware firmware enables handy transition to real implementations. MSPsim is part of the Contiki OS project and can be enhanced in cross-level simulation scenarios conducted under the COOJA framework (Eriksson, et al., 2009).

ii. Simulation language

All the underlying models developed under the MSPSim framework are implemented in Java programming language.

iii. Key features

MSPSim is a powerful emulator due to its even-based simulation kernel that enables accurate execution timing with low resource utilization, thus providing high simulation performance even in scenarios involving thousands of nodes. In order to achieve this functionality, MSPSim processes the simulation events based on two queues that address the events to the simulator scheduler according to their time criticality. The first queue includes the events concerning the internal components of the MSP430, such as the analog-to-digital converter, and are scheduled based on the CPU clock cycles. Secondly, events concerning external components, like the radio

transceiver, are scheduled based on a simulated high-resolution clock. Furthermore, MSPSim provides a modular design for the simulated mote hardware peripherals such as the sensors, communication ports, the radio transceiver and LEDs. Hence, the user is able to modify the mote abilities according to the specifications of the simulated application. Finally, a GUI is integrated and enables handy simulation designing, debugging and visualization of statistics like CPU utilization over different scenarios. Also an accurate graphical representation of the sensor board is presented and simulates its visual behavior, for instance the color and flashing of the on-board LEDs.

iv. Limitations

Similarly to COOJA, MSPSim is implemented in Java and thus its extendibility, is limited to the heap memory of the Java Virtual Machine (JVM).

3.6.9 Avrora

i. Summary

Avrora is an open source instruction level sensor network simulator with a cycle accurate behavior (Titzer et al., 2005). Unlike other simulators that are able to simulate only specific platforms, such TOSSIM, Avrora has language and OS independency due to the ability of processing actual machine code. The provided simulation and analysis tools are designed to emulate Crossbow Mica2 and MicaZ mote platforms, as well as custom motes that are based on the Atmel AVR microcontroller. Avrora simulates a network of motes that process real microcontroller programs, in preference to model-based abstractions so as to provide convenient evaluations. Additionally, the underlying components of the mote are simulated as individual software that interacts with the simulator core through respective virtual interfaces. Finally, the Avrora project was transferred to sourceforge on 2008 and its development has been discontinued since 2013 (Avrora - SourceForge.net, 2013)

ii. Simulation language

The Avrora framework is implemented in Java programming language in order to provide flexibility and portability.

iii. Key features

One of the key features of Avrora includes its accuracy on simulating the time model based on the clock cycle. Moreover, the host computer processes all the simulated nodes as individual threads that are further synchronized when necessary, in order to ensure global timing and communication order. Moreover, large-scale scenarios can be efficiently simulated with reasonable performance according to the number of the available processors. Additionally, in order to execute the subjected applications, the simulated events are processed based on an event-queue that takes advantage of the sleeping-mode property of the nodes, thus promoting performance efficiency. The developers of Avrora claim that it is able to scale networks of up to thousands of nodes.

iv. Limitations

A major drawback of Avrora is that it does not model clock drift, which is a situation that occurs when nodes may run at slightly different clock frequencies over time due to manufacturing tolerances, temperatures, and battery performance. Furthermore, Avrora lacks of a graphical user interface, thus conducting and analyzing simulations is a complex task. Finally compared to TOSSIM, Avrora does not support mobility and is 50% slower, however it provides more accurate and scalable evaluations.

3.6.10 Matlab

i. Summary

MATLAB (MATrix LABoratory), is a software package and a high–level scripting language which enables high performance numerical computation and visualizations of new ideas (MathWorks, 1994). It is the most popular software package for scientific research due to its powerful capabilities by means of analysis, flexibility and reliability. MATLAB provides a user-friendly environment that includes hundreds of reliable and accurate built-in

mathematical functions. These functions can be optimized and collaborate in order to provide solutions on a broad field of mathematical problems such as matrix algebra, complex arithmetic, linear and nonlinear systems, differential equations, signal processing etc. Therefore, MATLAB is particularly an appropriate tool to serve as a generic data-managing platform, as well as in the field of wireless sensor networks.

ii. Simulation language

The core functions and the build-in libraries are implemented in C and Fortran programming languages, however external libraries written in Python, C, C++, Perl, Java, ActiveX or .NET can be directly called from MATLAB.

iii. Key Feutures

The most important feature of MATLAB is its programming interface, which is very easy to learn and operate even from users with basic programming skills. Moreover, users are able to develop their own functions by using its native framework or by accessing custom libraries written in different programming languages through specific external interfaces. Furthermore, there are several optional toolboxes in order to support special application designs such as signal processing, control systems design, system identification, statistics, neural networks, fuzzy logic, symbolic computations, and so on. Additionally, MATLAB is able to simulate sensor networks by employing the modeling abilities of the Simulink framework, which is an integrated software package for modeling, simulating, and analyzing dynamical systems (Qutaiba, 2012). However, MATLAB provides an extendible design and is not limited only to Simulink, thus enabling researchers to implement different WSN simulation frameworks such as tinyLAB (Santini, 2009) and Prowler (Zhang et al., 2006).

Finally, MATLAB is the most famous framework for developing customized simulators aiming to study particular problems in WSN applications.

iv. Limitations

The only drawback of MATLAB is that it is an interpreted language thus resulting in slower processing performance. Moreover, MATLAB is a commercial software and requires users to pay in order to purchase the complete package.

3.6.11 EnergyPlus

i. Summary

EnergyPlus is an open-source next generation energy and building performance simulation tool, which was derived by combining the two well documented energy simulation engines of DOE-2 and BLAST (Building Loads Analysis and System Thermodynamics) (National Institute of Building Sciences, 2015), along with new capabilities. DOE-2 (DOE-2, 1998) is a popular freeware aiming to provide energy consumption predictions for buildings, while BLAST can be used to predict and analyze heating and cooling energy consumption within buildings. EnergyPlus provides a completely new and modular structure, where different modules can easily be included into the simulation so as to combine different concepts and aspects of building energy consumptions (Crawley et al., 2001).

ii. Simulation language

EnergyPlus was initially written in FORTRAN, a programming language for scientific supercomputing applications. However since version 8.2.0 Autodesk (Autodesk Inc., 2015), a leader in developing engineering software, has translated the simulation core, which comprises more than 700,000 lines of computer code, into C++ programing language (Roth, 2013).

iii. Key Feutures

The modular design of EnergyPlus is a key feature because it enables researchers to quickly add new modules to the program or even links to external programs. The C++ implementation provides the advantages of running on modern hardware like multi-core processors. Moreover, C++ is a popular and powerful programming language, thus increasing the accessibility of EnergyPlus to many more developers, who can customize their own programs. Therefore, energy management applications based on sensor networks can be also evaluated through the EnergyPlus simulation engine, similarly to the proposals of (Dong & Andrews, 2009), (Agarwal et al., 2010) and (Erickson et al., 2011).

iv. Limitations

A major drawback of EnergyPlus is that it lacks of a graphical interface and it can be accessed only from a console environment. Hence, simulations are mainly based on input files, which further increase the effort of defining all the necessary input data.

4. Testbeds

The growing interest of the research community in the WSN field of study imposes more accurate performance evaluation tools and methods. However, most of the WSN applications introduce significant challenges due to hardware availability, fluctuations of environmental radioactivity, resource constrains, energy autonomy, management, cost etc. Therefore, testing and verifying new designs, protocols and applications only over simulation may lead to inaccurate results, given the great complexity of real deployments.

This was a leading reason that motivated universities and research institutes across the globe, to implement and design experimental laboratories known as testbeds, so as to be able to reflect the exact environmental conditions that may face during real deployments. A testbed is a platform consisting of a number of low-cost and low-power devices known as nodes or motes, which are deployed in a controlled and manageable environment. Typically, the motes are equipped with sensors that communicate via wireless connection and monitor a phenomenon according to the testing scenario. Nevertheless, there are various testbed architectures based on their functionality and their modeling ability.

4.1 Testbed Requirements

An experimental physical testbed, in order to be able to address the diverse characteristics of WSN systems should be flexible in a way to support a number of different network topologies and protocols. Moreover the underlying infrastructure should enable the developers to test their solutions in the most realistic manner possible, by means of scalability, functionalities, environmental conditions and limitations. According to (Tonneau et al., 2014), the requirements of a WSN testbed can be classified into four main groups, which are the experimentation requirements, the hardware requirements, the mobility features and the maintenance considerations.

4.1.1 Experimentation Requirements

The experimentation requirements specifies the appropriate tools and actions that need to be supported by the testbed infrastructure in order to enable the users to design, conduct and analyze their experiments in a convenient and reliable way. These requirements include the experimental scenario specification, the communication interfaces, the experiment repeatability and simulation.

i. Scenario Specification

The first stage of the experimental process over a testbed is to indicate the appropriate resources such as the type of sensors and the number of nodes as well as the employed protocol stack, firmware and the data format. Therefore, the initial experimental setup of the subjected scenario is considered to be important in order to retrieve meaningful results.

ii. Interfaces

Communication interfaces enable the users to interact with the nodes and the other networking devices of the WSN testbed. Moreover, the researchers should be able to adjust and optimize the network parameters as well as to monitor and debug the ongoing progress of an experiment. It is of a great importance for the researchers to have access to network metrics, such as delay, throughput, overhead and energy consumption in order to be able to collect and analyze the resulted data.

Typically, there are two ways to access the testbed facility. The simplest type of access is to establish a Secure Shell connection with the use of the Secure Shell (SSH) protocol. However, modern laboratories employ web services to provide the appropriate interfaces for the users in order to interact with the network resources. Additionally, the web services technology provides the ability to the researchers to develop special client applications according to their needs. Nevertheless, this type of access introduces further security issues that should be taken into account.

iii. Repeatability

The researchers, in order to effectively evaluate their proposals, should be able to conduct a number of different experimental scenarios by varying

specific parameters, so as to investigate their influence on the overall system performance. Moreover the testbed facility must provide the ability to the users to implement their experiments independently to the underlying infrastructure so as to acquire more representative results. Therefore there are various methods that address experimental repeatability such as the standardization of the scenario specifications and the firmware of the nodes, as well as storing the traces of the experimental execution.

However, repeatability is still a challenging task due to environmental and system fluctuations related to radio interference, node mobility and hardware platform. For instance, environmental noise may cause instability between the links of the nodes that may further mask significant system events (Rensfelt et al., 2011).

iv. Simulation

As already mentioned, simulation-based validations for WSN applications lack of accuracy in capturing realistic environmental conditions, such as radio propagation. Nevertheless, there is a modern tendency among the research community in developing testbeds capable of combining both simulation methods and physical hardware experimentation. Such facilities benefit from the increased flexibility provided by the simulators and are able to test scenarios with high scalability (Coulson et al., 2012).

4.1.2 Hardware Requirements

As far as it concerns the hardware requirements, they play a critical role for the realistic performance evaluation of the subjected application. The testbed of choice should correspond to the appropriate hardware requirements of the application in order to enable the researchers to investigate in depth its functionality prior to real deployment. The hardware parameters of a physical WSN platform include the network heterogeneity, scale and federation.

i. Heterogeneity

Modern concepts that incorporate WSN technology such as IoT applications, rely completely on heterogeneous networks where the underlying devices play different roles and reserve various amount of resources. However, this application model introduces high complexity, thus requiring the testbed

platform to provide special designing and development tools that enable the researchers to conveniently conduct such experiments. Typically, heterogeneity can be distinguished into three types (Yarvis et al., 2005). First is the computational heterogeneity where some of the nodes have increased computational abilities such as the sink nodes and the gateways. Second is the link heterogeneity where some of the nodes may have wired interfaces in order to provide reliable communication links. Final is the energy heterogeneity where the nodes have various energy resources.

ii. Scale

Most WSN systems are deployed in large areas such as smart-city applications, where thousands of nodes are involved in the network. Hence, the researchers need to test their solutions in scenarios with increased scalability. However, most physical testbeds consist of only few nodes, from tens to hundreds, due to the high cost of developing such hardware. On the other hand, modern hardware technologies have significantly decreased in cost, thus enabling scientist to develop new physical experimental platforms or to extend existing ones, so as to be updated and address the contemporary challenges.

iii. Federation

Another method that is employed in order to address scalability and heterogeneity issues in WSN testbeds, is the federated model. This feature enables local experimental platforms to interconnect under a common framework in order to share their resources and provide more powerful evaluations. Therefore, the scientists are able to authenticate and reserve simultaneously the appropriate resources amongst several local testbeds that are members of the same federation (Chatzigiannakis et al., 2009). However, a major challenge of this concept is to maintain link reliability between the interconnected testbeds; thus requiring QoS models to be employed so as to ensure efficient real-time execution of the subjected experiment (Ricci et al., 2012).

4.1.3 Mobility Features

There are many WSN applications that involve mobile nodes, which require communicating so as to interchange sensory data and information. Hence, there is a need to develop experimental facilities that employ robotic and automation systems in order to enable researches to test such applications. However, the mobility feature in WSN testbeds introduces some important issues that need to be considered in order to effectively design and conduct experimental evaluations. These issues include the mobility type, power recharging, localization, designing and management.

i. Mobility models

Globally there are two types of mobility models, which are the undergone and the controlled mobility. By the term undergone refers to the mobility of a node that is attached to either an object or an entity, which cannot be controlled by the device itself. Moreover, as far as it concerns entities such as animals or humans, the mobility pattern is not possible to be predicted. On the other hand, in cases where objects such as public transport buses and trains that carry nodes, the mobility patterns are predictable since the routes are always predefined. However, considering the great complexity and cost of implementing robotic systems, only few testbeds support real-time experiments with mobile nodes. For instance undergone mobility has been implemented and supported by (Des Rosiers et al., 2011) and (Nati et al., 2013), as well as controlled mobility by (Jiménez-González et al., 2011), which introduces further management issues by means of locating and charging.

ii. Autonomous recharging & localization

Also in cases of hardware failure, mobility introduces further challenges considering the localization of the nodes. Moreover, the mobile components of the facility such as the robots should be able to locate the recharging point so as to promote the continuity of the experiment. Therefore, the WSN testbeds in order to provide autonomous experimentation should facilitate the process of maintenance by implementing accurate positioning and path planning mechanisms with obstacle avoidance.

iii. Designing and management tools

From the perspective of the user, the testbeds also should implement a number of tools that would facilitate the development and management of the mobile scenarios. These tools include software frameworks that provide hardware abstractions in order to enable the researchers to develop the appropriate embedded firmware and services. Additionally, visualization tools must be provided so as to define experimental parameters such as the paths of the nodes as well as to present experimental data in real-time.

4.1.4 Maintenance

Similarly to the general maintenance requirements that were described in previous Chapter, the WSN testbed maintenance is equally important in order to ensure the proper functionality of the system. Therefore a daily maintenance is appropriate to verify that the hardware and the software architecture are still operational so as address effectively the experimental queue. Additionally, scheduled maintenance must take place in order to update the provided services and hardware components such as the batteries, in a way to optimize the functionality and extended the lifetime of the facility.

4.2 Testbed Architectures

In a report that was conducted during the 2002 Workshop of National Science Foundation (NSF) (National Science Foundation, 2002), the authors analyze the experimentation process in the wireless networking research field. By adopting the NSF's perspective of analysis, we can classify the WSN testbeds into two groups, first based on their main objective and second based on their underlying structure (El-Darymli & Ahmed, 2012).

4.2.1 Objective-Based Classification

WSN testbeds can be distinct into two categories, based on their functional objectives, which are presented in Figure 13.

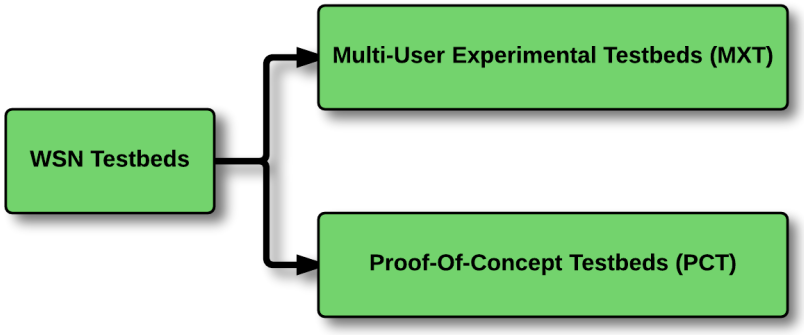

Figure 13. Objective Classification of WSN Testbeds (El-Darymli & Ahmed, 2012)

i. Multi-User Experimental Testbeds (MXT) – open testbeds

An MXT is designed to provide the research community with the ability of evaluating new network architectures, protocols and applications. The institution that manages the MXT is responsible to provide the researcher with access to its tools and its underlying infrastructure.

ii. Proof-of-Concept Testbeds (PCT) – custom testbeds

This kind of testbed is designed to advance and evaluate scenarios of specific research issues. New ideas are tested in a more constraint environment, in order to promote technology optimization. Moreover, the PCTs can be considered as the critical stage for the final commercialization. Ordinarily, if the concept is proofed then the PCT is no more of use.

From the above stated reasons, it is obvious that the missions of MXTs and PCTs are quite different. An MXT provides its services to a wide range of users, whom research can be focused in various issues. By contrast, the PCTs aim in a particular research objective. Choosing the suitable type of testbed, in accordance with the nature of the problem, can play a crucial role for conducting optimum performance evaluation tests.

4.2.2 Structure-Based Classification

Furthermore, WSN testbeds, based on their structure, can be categorized into four groups that are relative to each other. The most simplified structure of WSN can be considered the Research Kit (RK). Next, the Cluster Testbed (CT) provides a broad infrastructure that employs similar elements with the RKs. Overlay Testbed (OT) is overlaid on an existing testbed, which can be irrelevant to WSN technologies. Last, but not least is the Federated Testbed (FT), which is a wider class and is able to involve and combine all the previous structures. Figure 14 illustrates this affinity. Further description of the above-mentioned categories can be found below.

i. Research Kit (RK)

A WSN Research Kit is a pack of WSN software and hardware, produced and developed by various vendors. The relatively low price of an RK combined with its convenient installation, render a suitable solution for researchers to built their own local testbeds.

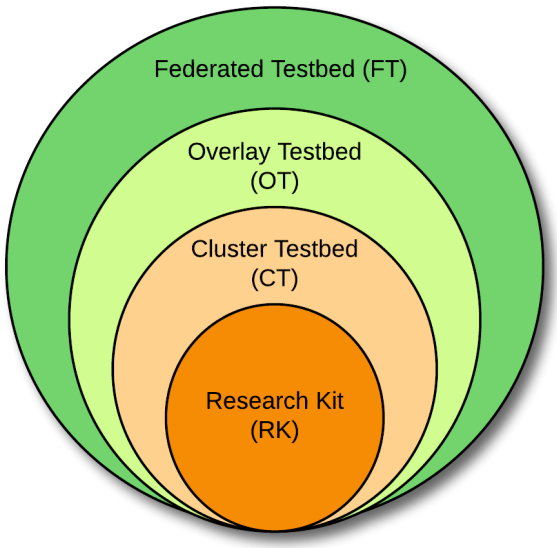

Figure 14. Structured Classification of WSN Testbeds (El-Darymli & Ahmed, 2012)

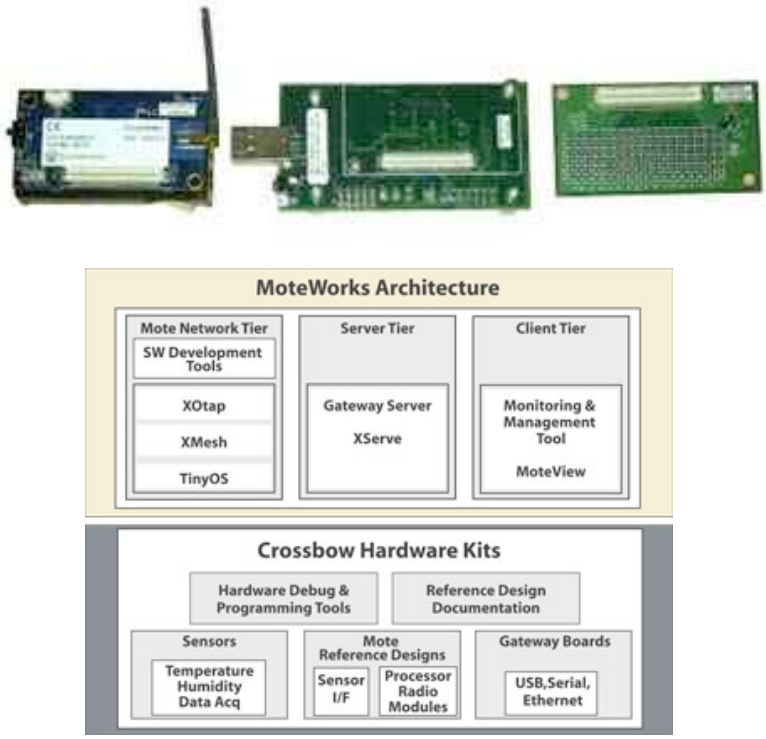

Figure 15. Crossbows' RK and its SW Platform

Due to the great diversity of research applications, the vendors develop different RKs with the appropriate components. However all the RKs have a common basic structure, consisted by a number of nodes and base stations. The nodes have sensors such as light, temperature, humidity, motion etc. and a programmable firmware in order to implement their functionality. They can be powered either over USB or batteries. Monitoring and management software is also provided, like the MoteView by Crossbow. Finally, with respect to scalability, the vendors have developed software platforms to support this feature, like Crossbows' MoteWorks as depicted in Figure 15.

ii. Cluster Testbed (CT)

A Cluster Testbed (CT) is an experimental laboratory, enabling researchers to perform extensive evaluations of their solutions in large scale, over real deployment emulations. It can be accessed remotely only by authorized users. The majority of existing testbeds belong to this category. Regardless the different characteristics of a CT, it should be flexible into adopting various configurations. In other words, the users should be able to experiment on modular network architectures and topologies, in order to control certain

features, such as scalability, power consumption, transmission power, etc. Hence, many CTs are designed to be open and expandable. Moreover, the CTs should have a support team for maintaining and troubleshooting the undelaying infrastructure, so as the researchers can focus on the experimentation process of their study. Some examples of CT are Motelab, TWIST, Indriya, Mirage, NetEye. Even thought there is not any specific framework for developing a CT, the majority shares similar interconnection model between their components. Those typical architecture scenarios are illustrated in Figure 16, followed by a brief description of the underlying elements.

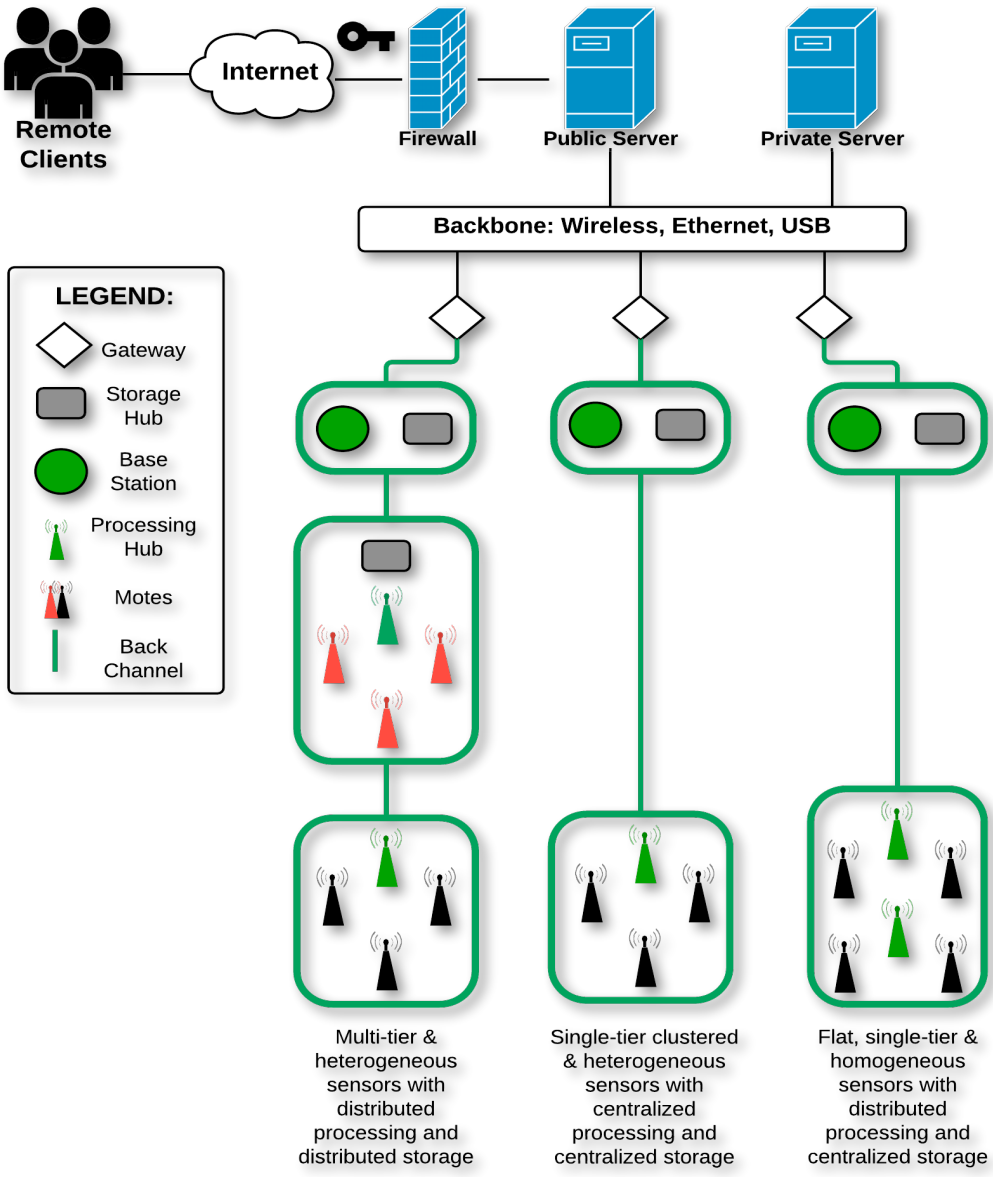

Figure 16. Typical WSN-CT Architecture Scenarios

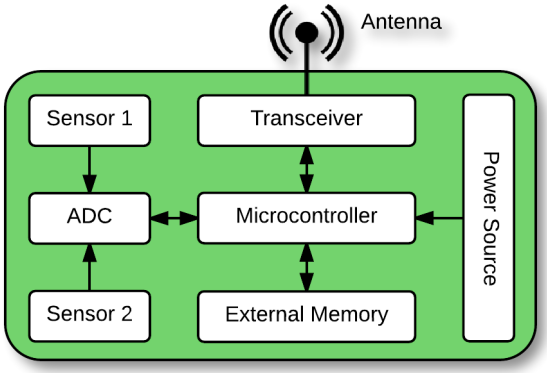

Figure 17. Typical Sensor Node Structure

A. Sensor Nodes – Motes

WSN motes are small, low-cost, wireless electronic devices that are capable of gathering and processing sensory data, like temperature, humidity, motion, pressure etc. There are various vendors that produce different types of motes with various capabilities. However, almost every mote employs similar components. Figure 17 depicts a generic internal structure and interactions between the elements of a node.

The sensors can be categorized into active and passive, based on their probing ability. An active sensor probes continuously the environment within its range, like sonar and radar. On the other hand, a passive sensor collects its measurements without actually manipulating the environment by active probing. Most of the researchers choose passive sensors for their solutions. However, any sensor produces analog signals that are further translated into machine language, through a conjunction of the ADC and the microcontroller. Moreover, the microcontroller is responsible to perform all those tasks, between the internal components, that are critical for the overall functionality of a mote. Additionally, there is a memory to store the required flash data so as to program the node, as well as application related data. In order to communicate with neighbor nodes, data exchange can be achieved through the transceiver. Ordinarily, communication and processing functionalities consumes the most energy. Nodes have low power source that can be either a battery or an energy harvester, thus raising a constraint for adopting energy efficient protocols.

Finally, according to the abilities of a node, we can classify node deployments in two groups. Homogenous sensor nodes are those that share similar

abilities, like transmission range, memory capacity, computing. Hence, heterogeneous sensor nodes have different abilities. Both groups can be deployed hierarchal in multiple layers or blended in a unified layer.

B. Processing Hub (PH)

Taking into consideration the small size of a node, it is obvious that its recourses by means of storage, processing and energy are limited. Thus, Processing Hub emerged to content the need for a more powerful mote. The PHs are more expensive than the common nodes due to their expanded sufficiency, and can be assumed as base stations. In order to advance the network resource utilization, the PHs collects sensory data from the other nodes and produce compact information that is further promoted to the network.

C. Storage Hub (SH)

As it was stated previously, sensory data from nodes are transferred to central base stations for additional processing due to storage constrains. Nevertheless, there is a necessity to point out some important events before data reaches the end user. This feature can be implemented by deploying an SH, which by its side utilizes data mining and feature extraction software tools. The presence of SHs in WSN testbeds is not obligatory.

D. Gateway (GW)

The Gateways act as the last step in the information route, by bridging the sensor network with the rest of the network. It is an IP addressable component, and aggregates data from the base stations to the servers and vise versa.

E. Back-Channel (BC)

The back channel is a critical element for optimum performance and maintenance of any WSN testbed. It is the data transmission medium between the gateways and the sensor network, enabling node programming, monitoring, data logging etc. The BC can be either wired or wireless, both having their pros and cons. The wired BC is used extensively through indoor sensor deployments, due to handy use of USB or Ethernet cable channels.

Supporters of wired back-channel assert that utilization of such technology can avoid network congestion in the wireless channel, in order to be devoted only to application related traffic (Handziski et al., 2006), (Werner-Allen et al., 2005). On the opposite side there are some researchers claiming that adopting wired models leads to impractical solutions (Dimitriou, Kolokouris, & Zarokostas, 2007).

F. Back-Bone (BB):

The BB is the infrastructure of the testbed that interconnects various elements of the network. Typically, it provides a communication path between the gateways and the servers of the network. Various technologies can be used to implement a BB, either wireless or wired Ethernet and USB. The researchers choose between wired and wireless solutions based on the WSN scalability and location. Ordinarily, indoor testbeds employ wired technologies, while outdoor WSNs utilize wireless solutions. Sometimes designers deploy the back-channel as part of the back-bone.

G. Private and Public Servers

Servers host a number of software tools that are applied on a central database, containing information about the WSN testbed. The database takes on an intermediary role between public and private servers. The remote users place requested tasks on the public server that are further retrieved and processed by the private server. Moreover, the database interacts with the sensor network in order to be updated with the current status of the testbed elements, as well as storing logging and localization information about the motes. Additionally, the servers should provide an interface so that the end users can log on and access the database and its tools. Finally the servers communicate with the base stations and the motes, through the gateways that are connected to the backbone.

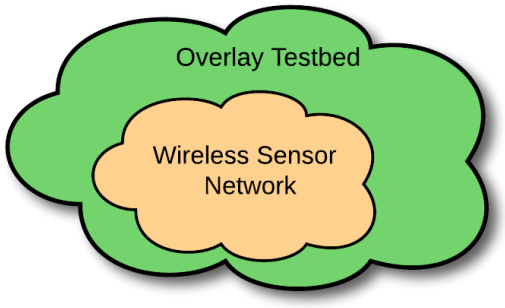

Figure 18. Macroscopic structure of an OT

iii. Overlay Testbed (OT)

The OT follows the same concept as the Internet being overlaid on the telecommunications network (Figure 18). In the research community of wireless communications, OT is widely accepted as the most efficient tool for evaluating new protocols and applications, because it allows explicit investigation of unforeseen network models. During the deployment of an overlay testbed, the underlying network is not affected. Typically, OTs are developed to experiment wireless communications in general, hence they may not have been designed for testing sensor networks. However, there are cases that a WSN is overlaid by a broader testbed.

The structure of an OT is a combination of the underlying infrastructure of the overlay testbed, with the comprised elements of the internal sensor network. Moreover, the structure of the overlaid WSN follows similar principles with previously explained cluster testbeds. A famous example of an OT is the early Emulab. However, the evolution in the network technologies has expanded the concept of OTs to virtual federated testbeds.

iv. Federated Testbed (FT)

The development of a federated testbed can be achieved by interconnecting various locally manageable testbeds, in different geographic locations. Those individual laboratories are connected through the Internet, in order to frame a broad platform that enables vast experimentation of new applications.

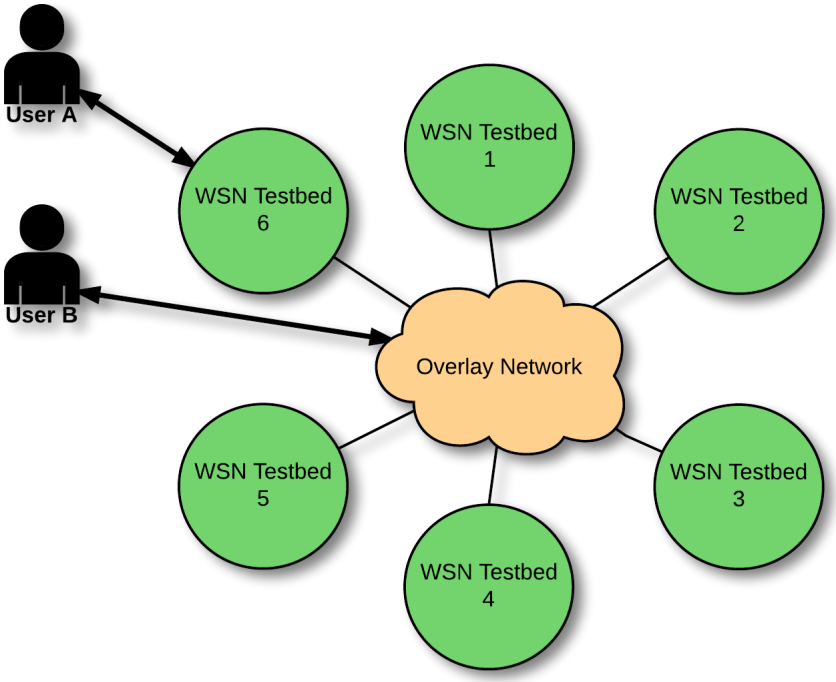

Figure 19. Typical Federated Testbed model

Due to the great variation in the underlying infrastructure and the complexity of the employed network architecture of an FT, researchers can investigate unpredicted issues that may occur. Furthermore, the heterogeneous hardware diversity, along with the unlimited capabilities by means of recourses, promotes cost efficiency in WSN research projects. Hence, any experimental model can be adopted conveniently, without facing the constraint environment of a single facility. However, a generic architecture can be modeled and is depicted in Figure 19.

The FTs' structure is an ensemble of the underlying individual WSN testbeds that communicate through a federal overlay network. The member laboratories are developed and maintained by different institutions and their architecture is equivalent to the CTs described above. Each testbed owner has full jurisdiction over its own facility and allows authenticated local access by utilizing a web server. The overlay network is equipped with a member portal server that connects the individual testbeds using the Internet, in order to be perceived as a whole by the end user. This feature enables the researchers to experiment over a distributed virtual testbed with modular capabilities and resources. Such innovative network technology offers boundless research opportunities for new solutions. Examples of federated WSN testbeds include, FIT-IoT Lab, modern Emulab, WISEBED.

4.3 Survey of WSN Hardware Motes

As already described in the previous Section, the key component of any sensornet is the employed node platform. Moreover the actual hardware that hosts the application determines the basic capabilities of the application as well as the overall efficiency of the system performance. Hence the researchers are able to choose between various hardware platforms according to the requirements of their application. By taking into account the results of our research concerning the popularity of the motes as presented in Chapter 5, a brief description of those motes are summarized in Table 3 and can be introduced as follows.

Table 3. Summarized motes characteristics

Mote	Microcontroller	Antenna	Power	Sensors and I/O Boards
TelosB	TI MSP430	Internal	Batteries or USB	External sensors and I/O peripherals
TmoteSky	TI MSP430	Internal	Batteries or USB	On-board humidity, light, temperature sensors and external optional I/O peripherals
MICA2	Atmel ATmega 128L	External	Batteries or external source	External sensors and I/O peripherals
MICAz	Atmel ATmega 128L	External	Batteries or external source	External sensors and I/O peripherals
USRP	Software radio system	Up to 8 External	External power supply	Up to 2 transceiver I/O daughterboards
WARP	FPGA Xilinx Virtex-6 chip	2 external	External power supply	Optional I/O add-on boards
iMote	ARM7	Bluetooth external	Batteries	Optional I/O components

Mote	Microcontroller	Antenna	Power	Sensors and I/O Boards
IMote2	PXA271 XScale CPU	Internal and optional external	Rechargeable batteries or USB	Both side connectors for I/O peripherals
IRIS	Atmel ATmega1281	External	Batteries	Optional I/O peripherals
EPIC	MSP430	External	Battery inputs or external source	68-pin chip USB and Storage modules
FireFly	Atmel Atmega32L	Internal	Batteries	On-board light, audio, temperature, dual-axis acceleration and passive infrared motion sensors and optional I/O peripherals
Fleck	Atmel Atmega 1281	External	Battery set in addition to an inbuilt solar charging circuit	On-board temperature sensor and optional I/O peripherals and analog screw terminals
TinyNode	MSP430	Internal, optional external	Batteries or external source	On-board temperature sensor and an expansion connector for the SEB board

4.3.1 TelosB

The TelosB (MEMSIC, 2004) mote is an open-source platform designed by UC Berkeley to support the development of low-power research experimentation, which can deliver fast wake-up from sleep mode and thus extending the battery lifetime. Moreover, the mote is equipped with a USB interface that enables the researcher to program and communicate with the hardware without consuming battery energy, since it can be powered from the host computer. However, if TelosB is always connected thought the USB port, there is no need to load a battery set on the node. Also the Texas Instruments

MSP430 microcontroller of the mote is designed with an extended memory in order to be compatible with TinyOS applications as well as to interface with an optional sensor suit. Finally, TelosB communicates wirelessly through an integrated antenna that is connected to an IEEE 802.15.4 radio chip and is able to store up to 1 MB of logging data in an external flash storage.

4.3.2 Tmote Sky

The Tmote Sky (MOTEIV, 2005) was developed in replacement of the TelosB by UC Berkeley so as to increase the performance, functionality and expansion capabilities of the mote. It has on-board humidity, light and temperature sensors that further increase the hardware robustness as well as minimizing the cost and the size of the device. Additionally, its IEEE 802.15.4 radio chip enables the on-board antenna to communicate with high data rate within a rage of 128 meters from the node while providing link-layer hardware encryption and authentication. Finally, Tmote Sky employs an MSP430 microprocessor that supports TinyOS and later ContikiOS applications and is designed to load a protected OS image from the flash memory so as to recover in case of application failure.

4.3.3 MICA2

The MICA2 (Crossbow, 2003) mote is a commercial battery powered WSN hardware platform based on the Atmel ATmega 128L microprocessor. It supports TinyOS applications that are stored in an internal flash memory along with the communication protocols. Moreover, the mote is equipped with a radio chip that is compatible with 868/916MHz, 433 or 315MHz protocols and requires an external antenna in order to communicate with neighbor nodes. The device also provides expansion connectors and analog inputs so as to connect a wide variety of external sensors and peripherals such as serial or parallel interfaces that facilitate the programming of the hardware.

4.3.4 MICAz

The MICAz (MEMSIC, MICAz Datasheet, 2004) mote, similarly to its predecessor is a battery powered WSN hardware based on the Atmel ATmega 128L microprocessor. Moreover the microprocessor runs applications that are developed under MoteWorks, which is a TinyOS-based

framework. The employed radio transceiver is compatible with the 2.4GHz IEEE 802.15.4 standard and requires an external antenna so as to be wirelessly accessible. MICAz also provides expansion connectors and analog inputs to connect external sensors and peripherals such as Ethernet or USB interfaces that facilitate both programming and data communication.

4.3.5 USRP

The Universal Software Radio Peripheral (USRP) (Ettus Research, 2010) is a family of wireless hardware platforms that enable fast prototyping of flexible software radio systems. Moreover, the designing of the software modules is implemented in GNU Radio, which is an open-source software radio and signal processing package. Hence, the user after installing the GNU Radio software on his computer is able to communicate with the USRP hardware through either a high-speed USB interface or a Gigabit Ethernet link. Additionally, there are some USRP models that integrate a microprocessor capable of providing the appropriate functionalities in order to provide a standalone solution. Generally a USRP platform requires an external power supply and consists of a basic motherboard and a modular front-end that can accommodate up to two transceiver daughterboards with the corresponding external antennas. This modular approach enables the researchers to experiment with a great diversity of Radio Frequency (RF) up to 5.9 GHz.

4.3.6 WARP

The Wireless Open-Access Research Platform (WARP) (Rice University, 2006) is a scalable and extensible programmable wireless hardware that enables prototyping of advanced wireless networks. Its great advantage is that it combines a high-performance hardware suite along with an online open-source repository containing reference algorithms and support documentation, which is updated by the research community. The first two versions of the platform were developed by Rice University, however Mango Communications Inc. released the latest version on 2012 and ever since is the most active contributor of the project. The WARP hardware is powered over an external supply and consists of a main Field-Programmable Gate Array (FPGA) based on Xilinx Virtex-6 chip, thus offering a flexible way to implement different components of a wireless transmission system on various networking levels. Moreover the platform integrates two programmable RF

interfaces with external antennas as well as a variety of peripherals including an SD card slot, two Gigabit Ethernet interfaces, a USB port and other user I/O features. However the Ethernet interfaces are employed only to send and receive data traffic to the host computer. Therefore, the researchers are able to program the platform through its USB port. Finally, Mango as well as third party vendors develop a number of optional I/O add-on boards that extend the basic capabilities of the platform.

4.3.7 iMote

The iMote (Intel Mote) (Kling et al., 2004) is a hardware developed by Intel Reserch Labs in order to provide to the research community with a sensor node platform that is equipped with increased CPU performance, improved radio bandwidth and reliable. Moreover the hardware is powered over a battery set and consists of an ARM7 microcontroller, a wireless Bluetooth radio chip with external antenna, RAM and Flash memory as well as a number of optional user I/O components such as a USB and serial interfaces. The basic Bluetooth protocol was modified in order to meet the WSN specifications such as the “scatternet” mode of Bluetooth, which has been successfully adapted in order to be able to form networks of multiple piconets. Furthermore, networking and routing functionalities have been implemented on top of a TinyOS base in order to provide multi-hop networking and self-organizing abilities.

4.3.8 IMote2

The IMote2 (Adler et al., 2005) is an advanced wireless sensor node platform developed by Intel Research Labs as a replacement of its predecessor iMote. The structure of the platform consists of a low-power PXA271 XScale CPU running TinyOS as well as an IEEE 802.15.4 radio chip that is connected either to an integrated antenna or to an optional external antenna. Furthermore, IMote2 provides a modular design with interface connectors in order to enable the researchers to easily connect expansion boards on both sides of the board. Moreover, the top connectors provide a standard set of I/O interfaces for basic expansion boards. The bottom connectors provide further high-speed interfaces for application specific I/O in addition to a mini USB

port. The mote can be powered either through the USB port or by a battery board, which can be connected to either side. Furthermore, a special battery board can be employed in order to provide the option of mounting rechargeable batteries.

4.3.9 ZigBee-based Motes

ZigBee (ZigBee Alliance , 2002) is actually a standards-suite, which provides specifications for wireless communication protocols for PAN applications operating on small, low-power digital radios, rather than an actual hardware device. Moreover, the ZigBee protocol suite enhances and extends the IEEE 802.15.4 functionalities by providing low data-rates, low-power consumption, security and reliability due to the implementation of self-organizing mesh networking. Therefore ZigBee specifies a decentralized network topology very similar to the Internet that allows nodes to establish new routes through the network in cases of topology changes caused by system failures thus being a suitable solution for IoT applications. As far as it concerns hardware platform implementations, there are various vendors that produce ZigBee Certified devices and products that can be further employed by the researchers according to their needs.

4.3.10 IRIS

The IRIS (MEMSIC, 2011) is a 2.4 GHz battery powered mote module designed for low-power, wireless sensor networks. The mote structure consists of an Atmel ATmega1281 microprocessor and an IEEE 802.15.4 radio chip with an external antenna capable of yielding ranges as far as 500 meters without amplification. The microprocessor can support TinyOS applications and is able to load the MoteWorks framework from its internal flash memory. Moreover , IRIS is equipped with an expansion connector that enables the researchers to attach a great variety of optional interfaces and peripherals, including different sensor boards and a USB interface for both programming and data communication.

4.3.11 EPIC

The EPIC (Dutta & Culler, 2008) mote is a family of open hardware components developed by UC Berkeley as a solution for application-driven designs. General-purpose motes may introduce difficulties during the system development, since most applications require to be tightly coupled with the underlying hardware. Therefore, the modular approach of the EPIC mote enables the researchers to customize their hardware design by choosing the components that enhance the appropriate functionality. The EPIC family includes three individual components, which are the Core, USB and Storage. These components are compact multi-chip modules that can be conveniently integrated into new hardware designs through their 68-pin leadless chip carrier (LCC-68) footprint. Moreover, the Epic Core is a fully functional mote consisting of an MSP430F1611 microcontroller, flash memory and a radio chip that requires a power source and an external antenna. The Epic USB can be employed to support UART-over-USB, JTAG-over-USB, reprogramming, alkaline and Lithium battery inputs, Lithium battery recharging and automatic power source selection. Finally the Epic Storage provides a rich memory hierarchy of four different flash memories (NAND, two NOR, and FRAM), all with different read, write, and erase characteristics.

4.3.12 FireFly

FireFly (Mangharam et al., 2007) is a low-cost hardware platform developed by Carnegie Mellon University capable of providing both data processing and multi-hop mesh communication. The mote structure consists of an Atmel Atmega32L microcontroller that loads Nano-RK OS from a flash memory, as well as an IEEE 802.15.4 radio transceiver that is connected with an integrated antenna. Moreover, the hardware platform is equipped with various sensors providing light, temperature, audio, dual-axis acceleration and passive infrared motion measurements as well as I/O connectors that can enhance expansion boards including a USB interface and an AM receiver. Also there is an SD card slot for additional data storage. The AM receiver is employed in order to acquire the periodical synchronization pulses that are generated by a global AM carrier current transmitter.

4.3.13 Fleck

The Fleck (Sikka et al., 2007) motes are a series of WSN nodes developed by CSIRO ICT Centre aiming on outdoor applications where long range and energy self-sufficiency are crucial. The latest hardware platform from the group is the Fleck3B mote, based on an Atmel Atmega 1281 microprocessor capable of loading TinyOS applications from an 1Mb integrated flash memory. Moreover, it is equipped with a radio transceiver with an external antenna that can work in three different transmission bands including 433MHz, 868MHz and 915MHz. Furthermore, the mote provides a single connector for programming and access over serial ports, expansion connectors for enhancing Fleck daughter boards as well as screw terminals that enable convenient connection of analog and digital sensors; even though there is an integrated temperature sensor on board. Finally, the board requires 3.4-8V of power that can be supplied by a battery set in addition to an inbuilt solar charging circuit for NiMH batteries.

4.3.14 TinyNode

The TinyNode (Dubois-Ferriere et al., 2006) is an ultra-low power platform that provides a convenient way to add wireless communication to WSN systems. TinyNode consists of an MSP430 microcontroller optimized to run TinyOS from an internal flash memory as well as a radio transceiver that is connected to an integrated wired antenna with an optional connector for an external one. Moreover, the platform is equipped with an on-board temperature sensor in addition to an expansion connector, which is used to enhance the so-called Standard Extension Board (SEB). The SEB supports further analog, digital and serial interfaces along with the power supply that can be either external or a set of batteries.

4.4 Survey of WSN Tesbeds

After providing the aforementioned descriptions of the most popular mote platforms, it would be beneficial to continue with a brief survey of the experimental physical testbeds that were involved in our research as presented in Chapter 5. Therefore in this Section we will try to specify the key aspects and functionalities of the different experimental laboratories around the globe that can be employed by the researchers in order to validate their proposals. A brief summarization can be found in Table 4.

Table 4. Summarized testbeds characteristics

Testbed	Nodes	Key Features
MoteLab	Fixed array of 30 MICAz and 190 TelosB	One of the first open WSN testbeds. MySQL back-end server, a PHP web server, a Java-based data logger and a Job Daemon for assigning tasks to the motes. Wall-powered with in-situ power measurement device in addition to temperature, humidity and light sensors.
TWIST	102 TmoteSky and 102 eyesIFX	The testbed has a central PostgreSQL server and is hierarchically organized in three layers, the servers, the super nodes and the sensor nodes. USB powered with light and temperature sensors. The super nodes are Network Link Storage Units.
Indriya	139 TelosB	Based on MoteLab. The nodes are powered over the USB backchannel and equipped with light, temperature, acoustic, magnetometer, 2-axis accelerometer and infrared sensors.
Intel Mirage	97 MICA2 and 51 MICA2DOT motes	Based on a resource allocation system where the testbed resources are allocated according to a repeated combinatorial auction. The motes are equipped with pressure, temperature, light and humidity sensors and powered over Ethernet.
UMass DieselNet	40 buses with GPS devices and HaCom Open Brick computer with 3 radios	A vehicular DTN of 40 public transport busses and various throwboxes that work as relays that promote the messages to the central repository.
Emulab	580 PC nodes with USRP Mobile testbed with 6 MICA2 robots and 30 stationary MICA2	A combination of hardware and software tools. There are many instances of the Emulab framework deployed in more than two dozens sites around the world.
WARPLab	Up to16 WARP nodes controlled by a single PC	An experimental framework for experimentation of physical layer protocols by interfacing WARP nodes directly with MATLAB.

Testbed	Nodes	Key Features
FLOCKLAB	30 Observers equipped with any four of Tmote Sky, OpenMote, MSP430-CCRF, TinyNode, Opal, and Iris motes	A mixed indoor/outdoor topology, able to support different services such as measuring power consumption and time accurate tracing and actuation.
ORBIT	400 nodes WITH MORE THAN more than 1500 radio devices.	A radio grid network testbed that consist of a remotely accessible indoor testbed, in addition to an outdoor trial network with mobile nodes.
Tutornet	13 Stargates, 91 TmoteSky and 13 MICAz motes	A simple three-tiered, clustered WSN testbed
MAP	32 static mesh routers, 5 laptops and 16 PDAs	An experimental WMN laboratory. The testbed do not provide power consumption awareness.
NetEye	130 TelosB motes, 15 laptops	An open WSN experimental testbed equipped with light sensors and a mixed USB and Ethernet backchannel
KANSEI	210 stationary nodes equipped with a Stargate, a TmoteSky and an Extreme Scale Mote. 50 portable Trio motes and five robots	A heterogeneous, hybrid experimental WSN laboratory that combines hardware motes, simulation engines and data generation devices.

4.4.1 MoteLab

MoteLab (Werner-allen et al., 2005) was one of the very first fully functional and open WSN testbeds that was deployed at Harvard University in the three floors of Maxwell Dworkin Laboratory, the Electrical Engineering and Computer Science departments. The testbed provides a web interface that enables the users to easily create, manage and schedule experimental scenarios. Moreover, it automates the reprogramming of the motes as well as providing easy access to the testbed database that contains the generated data logs from the experiments. Additionally, the web services enable the

users to interact in real time with the nodes that are employed during an experiment. MoteLab consists of 30 MICAz and 190 TelosB motes deployed in a fixed array, as well as a MySQL back-end server, a PHP web server, a Java-based data logger and a Job Daemon that is responsible of assigning tasks to the motes. Moreover, the motes are wall-powered and equipped with an optional in-situ power measurement device in addition to temperature, humidity and light sensors. Finally every node is connected through an Ethernet interface to the back channel for convenient reprogramming and logging.

4.4.2 TWIST

The TKN Wireless Indoor Sensor network Testbed (TWIST) (Handziski et al., 2006) is an open, scalable and flexible indoor WSN testbed deployed in three floors at the Technical University of Berlin. The testbed structure is hierarchically organized in three layers, which are the servers, the so-called super nodes and the sensor nodes. The sensor nodes of the networks consist of 102 TmoteSky and 102 eyesIFX motes that are equipped with light and temperature sensors. The motes are powered over a USB interface that is also required for programming and further communication through USB hubs with the super nodes. The super nodes are Network Link Storage Units (NSLU) running a customized Linux OS while providing gateway functionality between the nodes and the servers. The servers and the control PCs are connected to the Ethernet back channel, thus requiring from the super nodes to be equipped with both USB and Ethernet interfaces. The control PCs are employed in order to manage and conduct the experiments as well as to support a central PostgreSQL server to store application and logging data. Finally, the testbed provides a web interface that enables the users to schedule and control their experiments.

4.4.3 Indriya

Indriya (Doddavenkatappa et al., 2011) is an open, large-scale, low-cost WSN testbed deployed at the National University of Singapore in three floors of the School of Computing. The users are able to access the testbed infrastructure through a web interface based on the framework proposed by the MoteLab engineers. Therefore the researchers can conveniently upload their programs to the nodes, create and schedule jobs as well as accessing the logging

information that are store in the central database after the completion of the experiments. Moreover, the testbed consists of 139 TelosB motes where most of them are equipped with light, temperature, acoustic, magnetometer, 2-axis accelerometer and infrared sensors. Additionally, the nodes are connected to the USB backbone that is used for programming and data logging as well as powering supply.

4.4.4 Intel Mirage

Intel Mirage (Chun et al., 2005) is a resource allocation system where the testbed resources are allocated based on a repeated combinatorial auction that is build over a closed virtual currency environment. Moreover, the users of the laboratory compete for its resources by submitting bids that correspond to combinations of interest in space and time along with a maximum value of virtual currency that the user is willing to pay. For instance a user's bid would be "any 32 MICA2 motes for 8 hours anytime in the next three days". Next, a combinatorial auction periodically collects the bids and specifies the winning users based on the overall availability and demand. The Mirage system was developed over a 148 node indoor testbed, deployed at Intel Research Laboratory in Berkeley. The testbed consist of 97 MICA2 and 51 MICA2DOT motes equipped with pressure, temperature, light and humidity sensors as well as an Ethernet interface that is used for power supply, programming and debugging.

4.4.5 UMass DieselNet

DieselNet (Soroush et al., 2009) is a vehicular Delay Tolerant Network (DTN) developed by the University of Massachusetts (UMass) deployed on 40 public buses that serve the surrounding area of the UMass Amherst campus. The DieselNet testbed is open to the research community for experimentation in addition to a number of stored traces that can be utilized for further simulation. Every bus carries a GPS device that records times and locations as well as a Linux-based HaCom Open Brick computer that is further connected to three radios; including an 802.11b/g Access Point (AP) to provide DHCP access to passengers, a second PCI-based 802.11b/g/a interface that constantly scans the surrounding area for DHCP offers and other buses, and a longer-range MaxStream XTend 900MHz radio to communicate with the so called throwboxes. The throwboxes are stationary wireless nodes that work as

relays and consists of a modified TelosB mote that is powered over a set of batteries that can be recharged by an attached solar sell. Moreover, the AP on each bus transmit its SSID every 100 ms. The second radio continuously scans for SSID broadcasts. On discovering a remote bus's AP, the discovering bus obtains an IP address from the remote bus. Then, a TCP connection is established between those buses, initiating a contact event, and data is continuously transmitted to the remote bus until the TCP connection is broken when the buses move out of range. Once the socket reports an error or closure, the contact event is marked as ended and logged. For each contact, the receiver logs the ID of the sender, the time, duration, and the number of bytes received. These bus-to-bus transfer records are transmitted to a central repository whenever a bus is able to associate with a throwbox.

4.4.6 Emulab

Emulab (Johnson et al., 2006) is a network testbed developed by the Flux Group and it is deployed at the School of Computing at the University of Utah. The testbed provides a combination of hardware and software that enable the researchers to experiment with a wide range of environments. Currently, there are many instances of the Emulab framework deployed in more than two dozens sites around the world. The scientists can access the facility without charge through a web interface that unifies the different environments and provides a more convenient solution for system evaluations and resource reserving. The laboratory consists of more than 580 PC nodes able to run any OS and emulate a great variety of systems and topologies. Moreover some nodes are equipped with USRP devices that enable the researchers to have the total control of the physical layer and its operations. Also many nodes are equipped with two 802.11a/b/g wireless boards that may act as access points, clients, or ad-hoc nodes and can be programmed through a wired interface. Moreover, a mobile testbed that is build over the Emulab network provides the ability to the researchers to experiment with six robots that are equipped with six customized MICA2 motes respectively in addition to 30 stationary MICA2 nodes deployed on the ceiling in a grid topology.

4.4.7 WARPLab

WARPLab (Anand et al., 2010) is an experimental framework that enables the researchers to rapidly prototype and evaluate new physical layer protocols by interfacing WARP nodes directly with MATLAB. Moreover, the baseband processing is performed within MATLAB while providing the ability to interconnect up to 16 WARP nodes that can be controlled by a single host PC. Every node within a WARPLab system consists of an FPGA board and a radio daughterboard with four large buffers respectively to the antennas. The FPGA handles the communication between MATLAB and the radios by transferring control signals and data between the host PC and the radio buffers. A typical WARPLab experiment cycle starts from MATLAB where the transmitted samples are first generated in addition to the baseband processing of the signal. Next, the processed signal is transferred over Ethernet to the radio buffers through the FPGA boards along with the appropriate control signals depending on the role of each node. Then MATLAB synchronizes all the nodes in order to start the experiment, where the transmitting node flushes its buffers through its radios while the receiving nodes immediately loads their buffers with incoming data. Finally, after the end of transmission the receiving nodes transfer the received signals to the host PC for further processing within MATLAB's interface.

4.4.8 FLOCKLAB

FlockLab (Lim et al., 2013) is a WSN testbed, developed and deployed at the Computer Engineering and Networks Laboratory at the Swiss Federal Institute of Technology Zurich in Switzerland. The testbed provides 30 powerful customized nodes in a mixed indoor/outdoor topology, able to support different services such as measuring power consumption and time accurate pin tracing and actuation. The customized nodes that are called observers are small Linux based computers that offers four target adapter slots to which different motes can be attached, including Tmote Sky, OpenMote, MSP430-CCRF, TinyNode, Opal, and Iris. Moreover, the testbed is organized in three tiers, where the lowest layer consist of the sensor motes that run the applications, the second layer consists of the observers computers that communicate over LAN or WLAN and transfer the data of the motes to the higher layer and vise versa. Finally, the third tier is a dedicated server that synchronizes all the observer nodes and provides basic functionalities such

as node configuration, experimental scenario management, as well as collecting, analyzing and visualizing data to the user.

4.4.9 ORBIT

ORBIT (Raychaudhuri et al., 2005) is a radio grid network testbed developed by the WINLAB research team at Rutgers University. It is an open facility that provides flexible, scalable and reproducible performance evaluations of next-generation wireless network protocols. Moreover, the ORBIT testbed consists of 400 nodes deployed indoors in a controllable radio grid structure, in addition to an outdoor trial network of vehicular and stationary nodes to support end-user validations in real environmental conditions. However, only the indoor deployment is accessible and programmable though a web interface, which provides various services that allow the user to interact with the testbed infrastructure in a convenient manner. Furthermore, the ORBIT nodes are customized hardware with a total of more than 1500 radio devices, including WiFi, WiMAX and LTE boards, USRP radios, WARP nodes, Bluetooth, ZigBee and TelosB motes. Therefore, ORBIT is capable of experimenting with end-to-end wired and wireless technologies through a common Ethernet back channel that is managed by a central server for equal resource sharing and data logging.

4.4.10 Tutornet

Tutornet (ANRG, 2009) is a three-tiered, clustered WSN testbed developed by Networked Systems Laboratory (NSL) and deployed at Ronald Tutor Hall at the University of Southern California, currently is managed by the Autonomous Networks Research Group (ANRG). Moreover, the laboratory is structured in three layers consisting of a central server, 13 Stargates version 7.3, 91 TmoteSky and 13 MICAz motes respectively. Every Stargate works as a cluster head and a base station by connecting seven member nodes through a USB hub. The Stargates establish an 802.11b connection with the server in order to interchange data as well as to reprogram the motes of the network. Additionally, the users benefit from a web interface that provides convenient communication with the underlying infrastructure of the laboratory.

4.4.11 MAP

MAP is an experimental wireless mesh network (WMN) testbed designed and deployed at the School of Electrical and Computer Engineering at Purdue University (Purdue University, 2008). Moreover, the testbed consist of 32 static mesh routers, which are connected through wireless links to each other, while providing 802.11b connectivity to end-hosts including 5 laptops and 16 Compaq IPAQ PDAs. Moreover, the routers are small computers equipped with a second radio in order to establish 802.11a/b/g connections between the nodes and communicate through a central gateway with the server of the network. However, researchers are not able to experiment with other multi-hop wireless networks such as sensor networks since the testbed do not provide power consumption awareness.

4.4.12 NetEye

NetEye (Ju et al., 2012) is an open WSN experimental testbed deployed in an office at the Computer Science Department at Wayne State University. Moreover, the testbed structure is organized in tiers consisting of 130 TelosB motes, 15 Dell Vostro1400 laptops and a central server. The nodes are equipped with a light sensor and a USB interface in order to be powered and communicate with the laptops though a USB hub. The laptops also work as cluster heads that host from 6 to 12 nodes each, while they are connected with the server through the wired Ethernet backchannel in order to be able to capture log data and reprogram the nodes. Also the server provides web interface so as to enable the users to easily create and mange experimental scenarios as well as to visualize the retrieved results.

4.4.13 KANSEI

Kansei (Ertin et al., 2006) is a heterogeneous, hybrid experimental WSN laboratory deployed at The Ohio State University that provides high fidelity with increased scalability. Moreover, Kansei is able to run experiments in large scale due to the combination of hardware motes, simulation engines and data generation devices. The basic structure consist of 210 stationary node array where each node carries three platforms; a Stargate a TmoteSky and an Extreme Scale Motes (XSMs) equipped with light, passive infrared, temperature and magnetometer sensors, as well as a microphone. The Stargate serves as a controller and a data collector for the TmoteSky and

XSM motes, which is connected through a dedicated 51-pin connector. Furthermore the Stargate is connected to the Ethernet backchannel in order to communicate with the central server. Furthermore, the testbed provides 50 portable Trio motes and five robots, each one equipped with a Stargate, a TmoteSky and a WSMote. Kansei has been designed to be highly fault tolerant, autonomic and self-organizing, thus ensuring the researchers to spend less time designing, conducting and troubleshooting their experiments.

5. Research

Ad-Hoc and Wireless Sensor Networks (WSNs) have enabled a large variety of applications. Environmental and wildlife monitoring, clinical medical and homecare monitoring, monitoring and control of industrial processes including agriculture, smart houses and cities are just some of the examples of Ad-Hoc and WSN applications, where low-cost, and easily deployed multi-functional sensor nodes is the ideal solution (Yick et al., 2008). As a result, during the last years we experience the emergence of a new paradigm called Internet of Things (IoT) in which smart and connected objects cooperatively construct a wireless network of things (Gluhak et al., 2011). However, the unique features of Ad-Hoc and WSN technologies can pose significant challenges. Hence, envisioned solutions must be verified before being deployed in a real-world WSN deployment, either by utilizing simulators and emulators or through experimentations by employing testbeds.

Simulation evaluation is an essential phase during the design and development of an Ad-Hoc or WSN infrastructure. However, environments in which Ad-Hoc or sensor networks evolve are often application-specific and too complex to be reproduced precisely. More specifically, simulators allow users to implement some basic assumptions, such as link quality, radio propagation, medium interferences and network topologies (Papadopoulos et al., 2013). Even tough, the majority of the simulation models cannot capture real world complexity, as proposed by (Hiranandani et al., 2013) and (Barrenetxea et al., 2008), they are often utilized as a first step. Our purpose is to show that this step is not sufficient to show the consistency of a solution as well as that low cost devices have steered researchers and engineers to enrich performance evaluation with testbeds.

Experimental evaluation is performed either custom or over open testbeds, and exhibits potential unexpected failures and problems that the proposed solutions by researchers would face during real deployments. Even though performing well over testbeds, those remain in vitro deployments with more or less controlled environment conditions. Such a proof of concept must then be transposed into the real world. Designing and setting up a complete Ad-Hoc or WSN system under real conditions that can support robust applications is a very complex task (Kdouh et al., 2012). Researchers and production system developers, first need an appropriate plan of deployment and later number of

tools, simulators/emulators and testing facilities for real experiments, in order to initially validate their concept or model and then to develop the appropriate infrastructure.

Throughout this study, we compile a large set of statistics on a literature review of 674 articles published in top conferences that are related with Ad-Hoc and WSNs over the 2008-2013 period. We focus on the evaluation provided by authors, and especially to what extent experiments on testbeds have become a must for performance evaluation of new communication algorithms and protocols. Hence, we exhibit the tendency where performance evaluation procedures rely on experiments with real hardware and environment, to the detriment of simulations. The question of scientific results versus proofs of concepts therefore arises. Indeed, we discuss the meaning of reproducibility and of a proof of concept as a prototype being designed to determine feasibility. In this paper, we also analyze the selection of the evaluation methodology (e.g. simulator, testbed), and simplicity of the overall design that should be provided for validation, understanding and explanation. Finally, this work aims to investigate and gather the pros both from simulation and experiments so that real- world experiments could lead to reproducible scientific results for our research community.

5.1 Performance Evaluation Procedures

In a typical research process cycle, once the modeling phase is done, network researchers and developers continue with the validation procedure in which they evaluate their concept by using either a simulator or an emulator. Later, network engineers and developers may proceed with experimentation to further cross-verify their proposal (Stojmenovic, 2008). Thus, once both the simulation performance and the experimental measurements are satisfactory then real deployments can be initiated.

5.1.1 Simulating protocols or experimenting algorithms

When facing too complex environments to be theoretically analyzed and considering the difficulties of setting up a real-world (e.g. large-scale) deployment, simulations used to emerge as the good mean to study Ad-Hoc and WSNs. Many open source and freely available simulators allow users to have a better control of the nodes by often employing a GUI, and to retain or simplify some assumptions in order to evaluate their solutions. Simulation

evaluation is a provisioning procedure during the protocol development. However, even if the simulation performance presents coherent results with mathematical analysis, past real-world deployments show that it is not recommended to proceed directly with real deployment since engineers may face unpredictable phenomenon such as node crashing or network disconnection (Barrenetxea et al., 2008), (Langendoen et al., 2006). Intermediate experimentation platforms can therefore be considered to bridge the gap between simulations and real world deployments. Nevertheless, simulations can offer wider sets of assumptions to test and potentially more complete evaluations. On the other hand, testbed experimentations do impose many characteristics, such as the physical environment, real hardware and network topology. Such facilities offer the opportunity to have their solutions facing real conditions, thus being more realistic than those modeled under software simulators. Yet, numerous parameters, including radio dynamics, link stability and symmetry, impact of the weather on communications (Boano et al., 2010), appear so unpredictable that they may lead to results that can not be reproduced with sufficiently tight confidence intervals. The ambition of obtaining scientific results should then lead researchers to allow for further repeatability of the presented results. As a result, during the simulation evaluation the environmental conditions should not affect the behavior of the nodes. Hence, it would be ideal if the authors first verify their model by employing experimental tests in order to reflect the reality that their proposals would face during real deployment.

5.1.2 A Thorough Literature Study

Throughout this Thesis, we carry out a thorough study over top representative conferences that are strongly related to Ad-Hoc and WSN research fields. In particular, we have studied all articles that have been published at the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), ACM Annual International Conference on Mobile Computing and Networking (MobiCom), ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc) and ACM Conference on Embedded Networked Sensor Systems (SenSys) conferences in order to derive the current tendency of the validation methodology that authors follow with respect to previously reported issues. Hence, we go through and study 674 articles in total published in the conference proceedings for the last six years from 2008 to 2013 where 596 are related to Ad-Hoc & WSN (see Figure 20).

Indeed, we identified 78 articles that deal with other wireless technologies such as WiFi and WiMAX, that are studied in the context of cellular networks. All of these papers have been found in MobiCom (i.e. 140 out of 185) and MobiHoc (i.e. 142 articles out of 175) conferences (see Figure 21), which are not entirely dedicated to Ad-Hoc and WSN but have a broader scope on mobility and wireless communications. We further emphasize our investigation over these 596 articles. During our investigation, we observe and obtain plethora of information for each work and later we categorize the articles based on their common features.

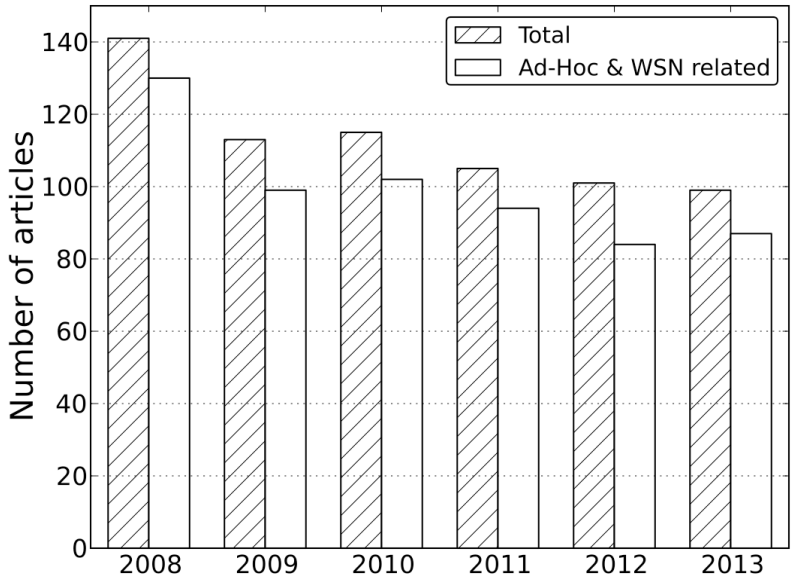

Figure 20. Number of articles per year (all conferences are considered)

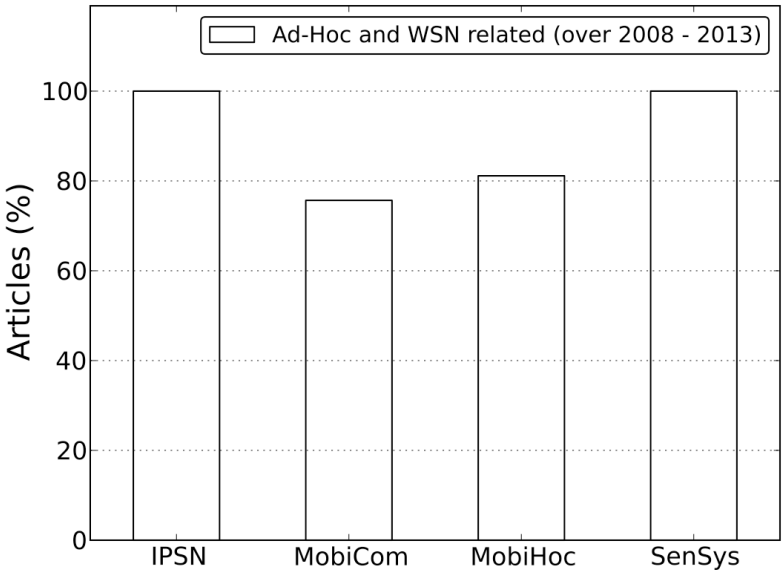

Figure 21. Appropriateness of our conference sample

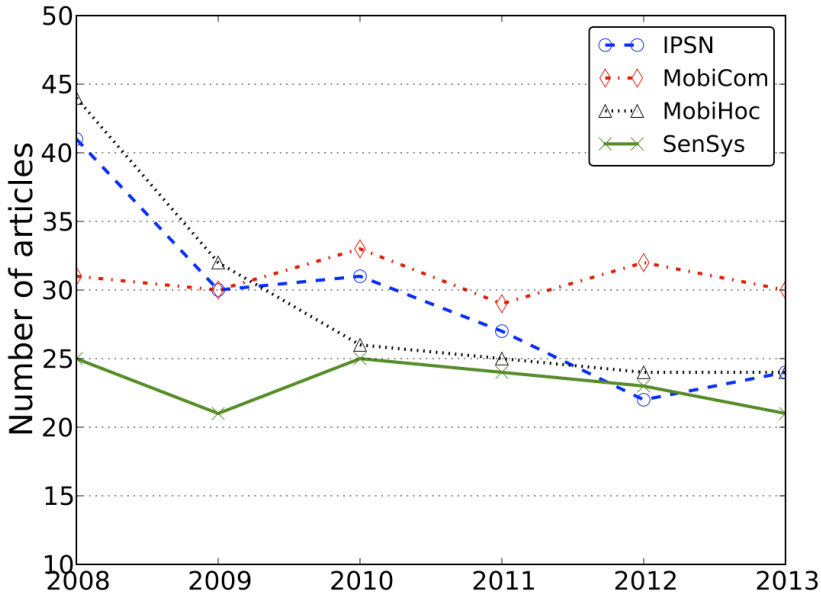

Figure 22. Publication flows over the period 2008 - 2013

Figure 22 provides detailed information about the total number as well as the Ad-Hoc & WSN related published articles per proceeding year. We actually observe that, there is a decreasing tendency of the published articles in the proceedings, indeed we identified 43 articles less from 2008 to 2013.

More specifically, MobiHoc and IPSN reduced the total accepted papers, from 44 to 24 (MobiHoc) and from 41 to 24 (IPSN) respectively, while MobiCom and SenSys kept a steady flow.

Modern technologies introduced the feature of mobility. Consequently, the research community focuses into developing and testing such aspects and scenarios. Our study results justify this trend, owing to the 148 articles (57.7%) that simulated mobile scenarios. Still, our statistical results for MobiHoc and MobiCom, the mobile oriented conferences, show that not all of their articles implement mobility scenarios. For instance, during the 2008 MobiHoc conference we determined only 13 out of 28 simulation-based articles that introduced mobility in their tests. As shown in Figure 23, 57% of articles having mobile scenarios are less induced by our conference sample (half of the conferences, MobiCom and MobiHoc, being theoretically focused on mobility-related topics) than by the global enthusiasm for mobile scenarios, all four conferences being considered.

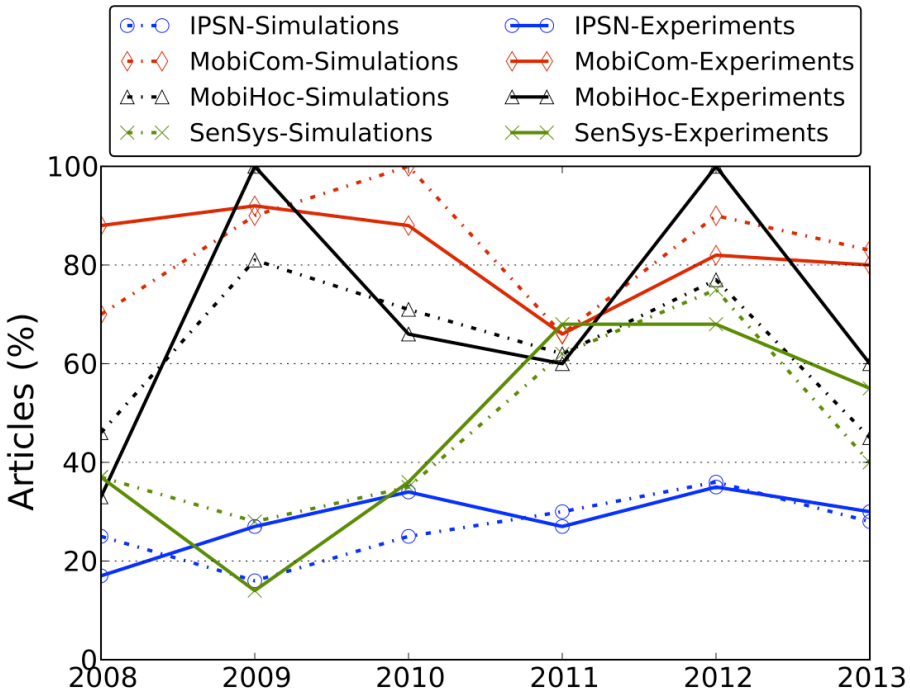

Figure 23. Mobility scenarios in performance evaluation procedures

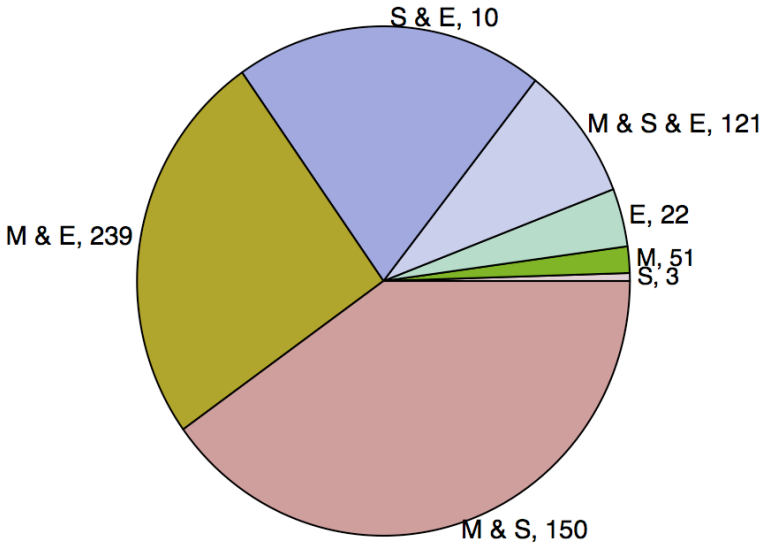

Figure 24. Use of Mathematics (M), Simulations (S), Experiments (E) and their combinations in validation procedures of 596 Ad-Hoc and WSN related articles.

5.2 Results Of Analysis

5.2.1 Evaluation procedures

In this subsection, we expose our analysis on the validation procedures that the authors followed. As a first step, we aimed to categorize the reviewed articles according to the employed evaluation method. In particular, we examine the proportion of simulation, experimental and mathematical (i.e. modeling or analysis) evaluated works. Our primary analysis exposes interesting results. More specifically, our investigation shows that the majority (i.e. 561) of the articles provide an analytical representation of their solution. The remaining 35 have only simulation or experimentation results. Furthermore, 284 verify their proposal by employing simulation evaluation while on the other hand 392 of the articles include experimental evaluation for their validation. Finally, only one out of five (i.e. 20.3%) articles proceed through all three phases of the research process cycle (i.e. analysis, simulation and experimentation). The number of articles with the previously stated properties (with respect to 596 studied papers) is illustrated in Figure 24.

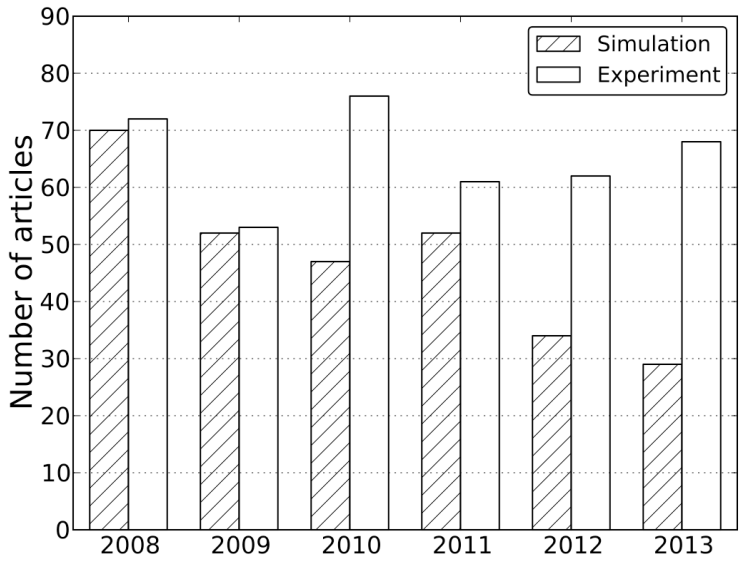

Figure 25. Total simulation versus experimentation evaluated articles.

We now present the characteristics of the articles that we studied. The percentage of simulation versus experiment-based studies (with respect to 596 studied articles) is illustrated in Figure 25. As can be observed, while simulations and experiments used to be equally until 2009, the usage of simulations is decreasing every year (except in 2011) while experimentations still remain present at a relatively stable rate.

Over the period 2008-2013, 284 studies followed a simulation-based evaluation to test their proposal. We noted the simulator usage, the scales of simulated networks and the programming languages used for custom simulators. Only 43.3% are validated through a known simulator while 42.3% of articles did not even provide any information about the tool that they have utilized (see Figure 26). Finally, 14.4% (with respect to 284 studied simulation-evaluated articles) developed a homemade simulator (Figure 26), by utilizing programming languages, such as Python and Java (see Figure 28) for the distribution of the most popular programming languages).

We are next interested in determining the usage of the simulators. As can be observed from Figure 27, MATLAB is the first choice in our community counting more than 35 articles, followed by TOSSIM, which has been utilized in almost 20 articles. Furthermore, Network Simulator 2 (ns-2) comes third with 13 articles.

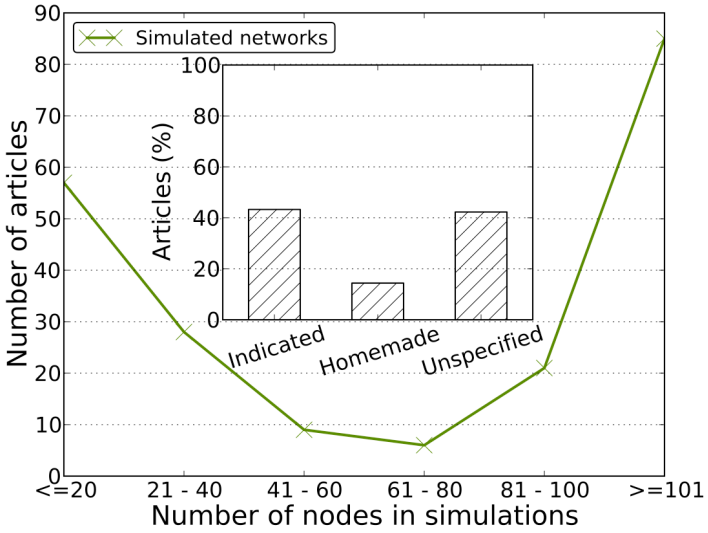

Figure 26. Simulator usage and scales of simulated networks

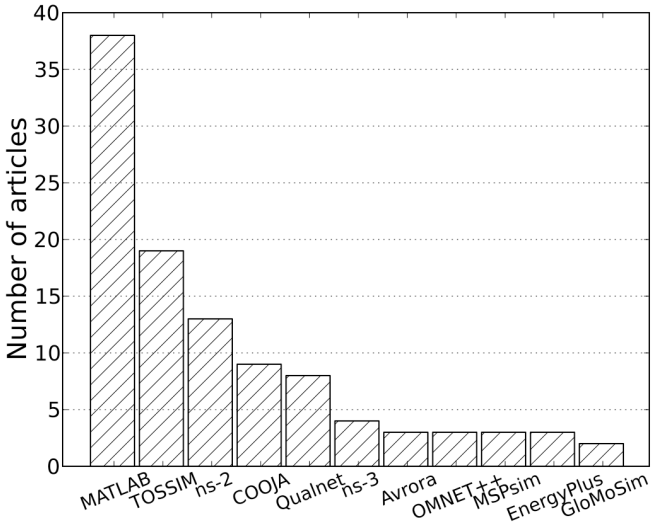

Figure 27. Popularity of simulators

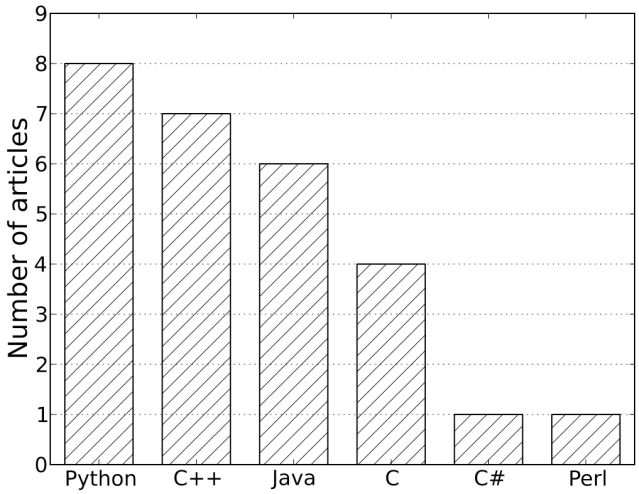

Figure 28. Programming language popularity for custom simulators

Nowadays, the research community is able to evaluate proposed protocols, models, and even new technologies over open testbeds at a very large-scale (Gluhak et al., 2011). Increasingly, network researchers are using experimentations to enlarge the scope of their performance evaluation, as it is illustrated in Figure 25. Moreover, as it can be observed from Figure 29, our investigation shows that the majority of the researchers, 91.3%, choose to set up their own testbeds. Even though to the current day, there are number of open facilities providing to the developers the infrastructure needed for experimental Ad-Hoc, WSN or IoT studies, only 10.7% of the articles use open platforms. Our compiled statistics tend to show that researchers would rather favor their own setups for small-scale deployments. In fact, among the 392 articles exposing experimental results, 78% of them do not exceed 40 nodes for their experimental setup (see Figure 29).

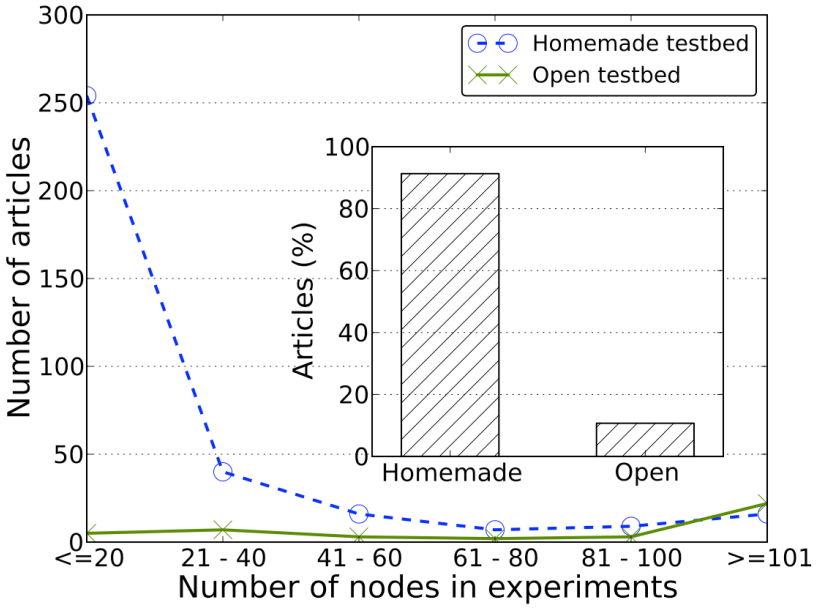

Figure 29. Testbed utilization and scales

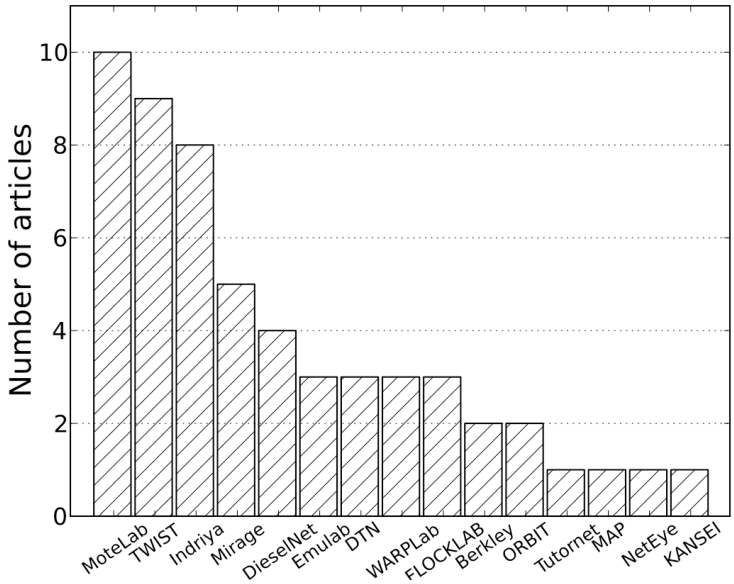

Figure 30. Popularity of open testbeds

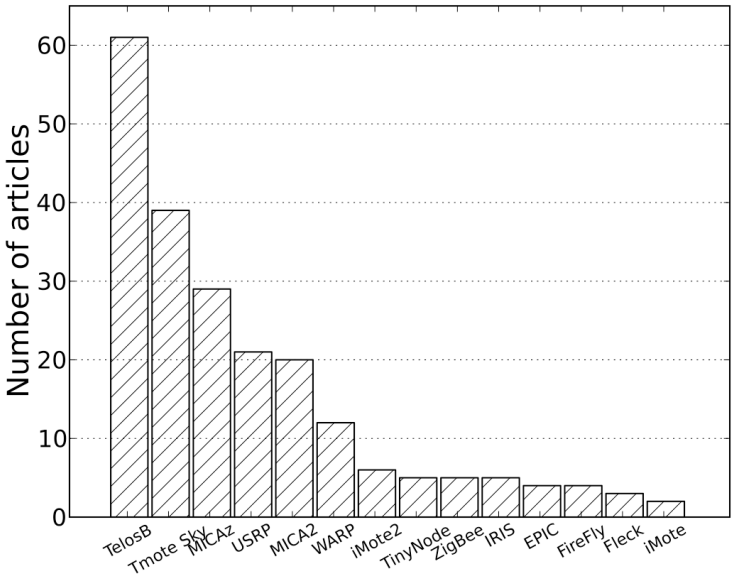

Figure 31. Motes Popularity

Hence, the increased difficulty to apprehend a remote open testbeds (e.g. specific hardware and software, network topology, booking procedure) may have induced researchers to set up their own relatively small scale networks.

Finally, we evaluated the popularity of the devices in homemade experiments. In Figures 30 and 31 the utility of the open testbeds and motes is presented. Even though a small number of articles experimented over open testbeds, we pointed out the popular open platforms. As observed in Figure 30, Harvard's

Motelab comes first (11 articles), followed by TWIST (10 studies). That can be simply explained as those facilities were the first to be open to the scientific community. Regarding the Indriya testbed, even though it was made available from 2011 only, it was used in 8 articles. The fact that users can interact with the testbed through the same intuitive web-based interface as MoteLab's could explain this success among the community.

5.2.2 Reproducibility

We continue our study by investigating the feasibility of reproducing results that are presented in the reviewed articles, both for simulation and experimental campaigns. To proceed so, we looked for some critical information (e.g. simulation setup, simulator indication, simulator details such as version or library, number of nodes), that should be provided by the studied articles. In order to reproduce the proposed solution, we assumed that the authors should provide a complete simulation or experiment settings subsection.

Regarding the simulation based evaluations, while only 43.3% of the articles indicate the simulator, 78.5% of those do provide some details about simulation setups. Among those, 72.5% precise the number of involved nodes. Finally, we decided of non-complete setups as soon as there was a lack of critical details regarding the tools used during simulations. For instance, as earlier discussed, MATLAB stands as the most popular software for simulations. In order to use it as a network simulator, researchers must import external libraries (e.g. as developed by the WISLAB1 team). It is difficult, if not impossible, to reproduce a simulation study when the version of a publicly available simulator is unknown, and only 21.5% provide us with the employed version or the utilized library of the simulator, which essentially concludes our outcome about the reproducibility of the simulation-evaluated articles.

We followed similar methodology for the experimental- based validations. Taking into account the nature of open platforms, the 42 articles, we consider that these articles in overall are reproducible. However, we counted 8 papers where the authors tested their ideas over both custom and open testbeds, with only 3 of them providing enough information to be assumed reproducible. On the other hand, the experimental results that are retrieved through homemade testbeds can be considered as difficult or even impossible to reproduce. This is explained since most of them are deployed in offices,

houses or even outdoor installations where the environmental radio activity varies, due to the interpolation of external features such as mobile phones, wireless routers and access points and so on. Nevertheless, owing to the nature (e.g. application layer) of the tested solution, we detected 31 homemade-based studies that may be reproduced. Finally, by summarizing the previous statements, we calculated that only 16.5% (65) of the experimental-based papers present reproducible results.

6. Conclusions & further discussions

6.1 Conclusions

Throughout this Thesis, we reviewed 674 papers that were published in four major and representative conferences in Ad-Hoc and Wireless Sensor Networks, over the period 2008-2013. We especially focused on the performance evaluation procedures in order to raise the question of whether simulations and experiments lead to scientific results or proofs of concepts. It is undeniable that simulators make the whole process of validation easier, faster and less expensive. On the other hand, with the growing development of open and realistic testbeds, researchers may overcome the technical challenges and economical barriers of real-world deployment to perform a thorough experimental evaluation of their ideas in wide- scale platforms. Simulators and open testbeds are two crucial and complementary design and validation tools; theoretically development process should start from the theoretical analysis by providing bounds and indication of its performance, be validated and verified by simulations and finally confirmed in open testbeds. Hence, once the entire procedure is successfully completed and the performance results show coherence, then researchers could promote their solution to engineers in order to proceed with real deployments.

Simulation evaluations should allow for reproducible setups, thus, producing scientific results that can be reproduced and verified by anyone in the community. In the context of experiments, our future work will focus on allowing researchers to get guidance for conducting experimentations over different testbeds, in order to cover larger sets of assumptions. Finally, as far as it concerns the specific issues studied by our research community, they can be considered a different approach from the one we followed, and binding those findings with our study, will also be a straightforward extension to our current work.

6.2 Further discussions

6.2.1 Scientific results or proofs of concepts?

Scientific results are expected to be repeatable while a proof of concept is a realization of an idea that demonstrates its feasibility. Our initial investigation shows that most of the authors choose to validate their proposals over experimental evaluation. Our investigation highlights some interesting tendencies in the networking scientific community, especially around Ad Hoc and Wireless Sensor Networks. As presented in previous Section, an increasing number of papers validate their proposals by using experimental evaluations.

We focused on the simulation and experimentation setups in order to determine if they were sufficiently described to allow for repetition of the evaluation procedure. While (Kurkowski et al., 2005) had focused on MANET, thus, looking for simulation parameters specific to mobility (e.g. speed of nodes, speed delta, pause time, pause delta), we aimed at a larger scope by gathering various sets of setup parameters. This is especially true for all observed experimentations among which setups are highly different (e.g. hardware, physical topologies, radio environment). The reproducibility level of experimental studies is lower than the simulation one. This is even more dramatic as this latest has not varied much since the study of (Kurkowski et al., 2005). More specifically, the authors had identified 29.8% of the simulation-based articles that did not identify the simulator used in the research. As already mentioned, for the 4 conferences we observed and over the 2008-2013 period, this proportion has raised to 42,3%. In addition, they calculated only 12.1% of the articles where the simulator version was mentioned. Furthermore, the authors were concerned that more than 90% of the published results may include bias. As a result, they conclude that approximately 12% of the MobiHoc simulation-based results appear to be repeatable. In (Kurkowski et al., 2005), numerous pitfalls throughout the simulation lifecycle had already been observed. Those tendencies, as already highlighted by take away from the goals of making the research repeatable, unbiased, realistic, and statistically sound.

As previously observed, over the last six years, less and less papers have actually considered simulations during their performance evaluation process. Still, the simulation phase allows researchers to demonstrate that the main principles of their proposal are indeed effective, before implementing them over a testbed (Stojmenovic, 2008). However, in order that the users will be

able to continue their proof of concept validation, we can avoid the necessity that they have to get familiar with various simulators and testbed platforms. Emulators such as TOSSIM or COOJA were developed to bridge the gap between simulation and experimentation, by being very close to real embedded systems in terms of architecture compilation targets. In fact, by utilizing these simulators, the very same code remains unchanged over the transfer from simulation to experimental campaign.

We are coming to a tradeoff between realism and reproducibility. More specifically, on the one hand there are more published articles that are closer to real deployment while on the other hand the reproducibility level of the studies decreases. So far, the proportion of papers using experimentations that allow reproducing the conditions of an experiment remains very low (< 11%). Moreover, all those testbeds are highly different (e.g. hardware, physical topologies, radio environment) and each would require a specific guidance to allow for scientific results to be obtained.

In (Kurkowski et al., 2005), the authors proposed a simulation study guidance. If the enthusiasm for experiments in networking scientific papers is to be confirmed, we should also be able to establish such mandatory steps to ensure statistically sound results. The significant number of open access and large-scale testbeds that have been deployed over the recent years (Gluhak et al., 2011), provides appropriate tools and experimental facilities for researchers and engineers to perform real experiments in order to further analyze their protocols. Open testbeds allow users to easily deploy source code (that could be the same with the one of the simulator) on a sensor node and to flash it at no delay. Those open platforms thus allow for more rigorous, transparent and replicable testing of proposed protocols and models.

Researchers, by connecting remotely (e.g. via SSH) to one open platform, may set up and initiate an experiment by using the terminal. Hence, the previously reported simulators along with open testbeds, allow the research community to get a flavor of real deployments while maintaining a unique programming code. More importantly obtaining performance evaluation measurements over large-scale network (both for simulation and experiments) can be at no cost at all.

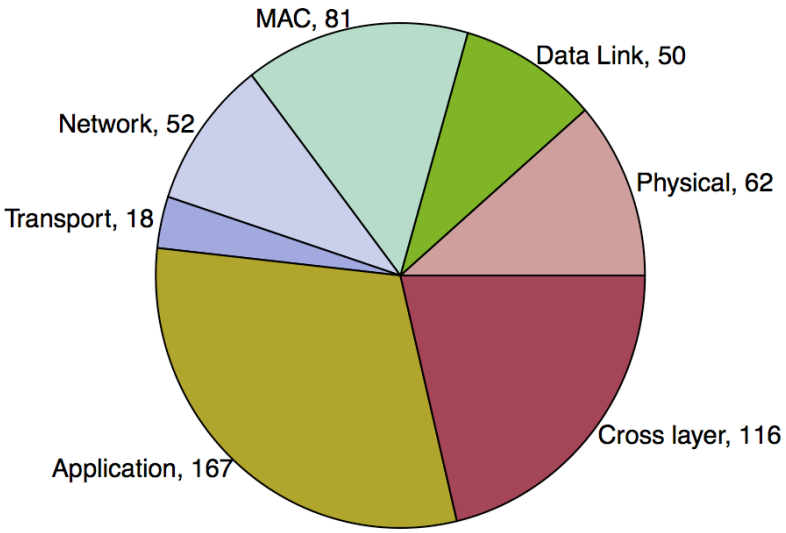

Figure 32. Main contributions of the 545 reviewed articles that include simulations or experiments

Finally, after following all the previously presented steps, and by obtaining coherent results, researchers may consider initiating a real deployment by utilizing their verified and refined protocol.

6.2.2 Applications

While studying the 674 papers, we could observe that the vast majority of papers actually mention some classical envisioned applications (e.g. defense, environment monitoring) but then focus on networking solutions that are application independent. Regarding the type and nature of the problems that were addressed, we collected data about the correspondence of the studied articles to the layers of the OSI model. We also identified papers that took into account some cross-layer design methodology.

As observed from Figure 32, the most common approaches were at application layer and with cross-layer design. While papers related to the former were investigating new kinds of information that could be collected by Ad-Hoc and WSN, contributions related to the latter were concerned with the high constraints imposed by low-cost sensor and mobile devices that impose to consider cross-layer approaches.

6.2.3 Mobility

Mobility is a key aspect for the future designs. While the majority of existing and used simulators allow to use and create mobility models, testing and executing such scenarios during an experimentation procedure requires to involve and combine advanced and intelligent technologies such as robots. Consequently, very few of the widely popular open platforms do support mobility (Tonneau, Mitton, & Vandaele, 2014). Actually, there are number of challenges that need to be addressed having mobile robots in a testbed, namely, charging, remote administration and maintenance of the robots. Indeed, robots must be able to reach their docking stations automatically. Conversely, remote users must be able to interact with robots over reliable links (e.g. WiFi). Even though those challenges can be addressed, testbed administrators then face the issue of localizing mobile devices in order to allow for repeatable trajectories. Indoor deployments can not rely on GPS solutions and thus impose distance approximations to be computed based on other available inputs (e.g. received signal strength intensity) or on costly technologies (e.g. 3D camera with range detector sensors for the mapping of the environment). Furthermore, even with perfect localization of all robots, trajectories would be very difficult to replay, especially due to the odometer drift. Some 3D cameras using range detector sensor aim at handling this drift. Still they lack to compute the path where not enough landmarks exist in open-space and large-scale environments.

Bibliography

Zeng, W., Chen, X., Kim, Y.-A., Bu, Z., Wei, W., Wang, B., et al. (2009). Delay Monitoring for Wireless Sensor Networks: An Architecture Using Air Sniffers. *Military Communications Conference (MILCOM 2009)* (pp. 1-8). Boston: IEEE Computer Society.

Zeng, X., Bagrodia, R., & Gerla, M. (1998). GloMoSim: A Library For Parallel Simulation of Large-Scale Wireless Networks. *Proceedings. Twelfth Workshop on Parallel and Distributed Simulation (PADS98)* , 154-161.

ZigBee Alliance . (2002). *ZigBee*. Retrieved 2015, from ZigBee: http://www.zigbee.org/

Zhang, Y., Simon, G., & Balogh, G. (2006). High-Level Sensor Network Simulations for Routing Performance Evaluations. *Third International Conference on Networked Sensing Systems (INSS06).* Chicago.

Zhou, H.-y., Wu, F., & Hou, K.-m. (2008). An Event-driven Multi-threading Real-time Operating System Dedicated to Wireless Sensor Networks. *Second International Conference on Embedded Software and Systems (ICESS 2008)* (pp. 3-12). Chengdu, Sichuan, China: IEEE Computer Society.

Dong, J. S., Sun, J., Sun, J., Taguchi, K., & Zhang, X. (2008). Specifying and Verifying Sensor Networks: an Experiment of Formal Methods. *Proceedings of 10th International Conference on Formal Engineering Methods (ICFEM 2008)* (pp. 318-337). Kitakyushu-City, Japan: Springer Berlin Heidelberg.

Wu, H., Luo, Q., Zheng, P., & Ni, L. M. (2007). VMNet: Realistic Emulation of Wireless Sensor Networks. *IEEE Transactions on Parallel and Distributed Systems , 18* (2), 277-288.

Wang, X., Wang, J., Zheng, Z., Xu, Y., & Yang, M. (2009). Service Composition in Service-Oriented Wireless Sensor Networks with Persistent Queries. *6th IEEE Consumer Communications and Networking Conference (CCNC 2009)* (pp. 1-5). Las Vegas: IEEE Computer Society.

Wang, C., Daneshmand, M., Li, B., & Sohraby, K. (2006). A Survey of Transport Protocols for Wireless Sensor Networks. *IEEE Network , 20* (3), 34-40.

Werner-allen, G., Swieskowski, P., & Welsh, M. (2005). MoteLab: A Wireless Sensor Network Testbed. *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)* (pp. 483-488). Boise, ID, USA: IEEE Comouter Society.

Wireshark. (2006). Retrieved 2015, from www.wireshark.org

Yarvis, M., Kushalnagar, A., Singh, H., Liu, Y., & Singh, S. (2005). Exploiting Heterogeneity in Sensor Networks. *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005). 2*, pp. 878-890. Miami, USA: IEEE Computer Society.

Yick, J., Mukherjee, B., & Ghosal, D. (2008). Wireless sensor network survey. *Computer Networks: The International Journal of Computer and Telecommunications Networking , 52* (12), 2292-2330 .

Autodesk Inc. (2015). Retrieved from http://www.autodesk.com/

Adler, R., Flanigan, M., Huang, J., Kling, R., Kushalnagar, N., Nachman, L., et al. (2005). Intel Mote 2: An Advanced Platform for Demanding Sensor Network Applications. *In Proceedings of the 3rd international conference on Embedded networked sensor systems (SenSys '05)* (p. 298). San Diego: ACM.

Agarwal, Y., Balaji, B., Gupta, R., Lyles, J., Wei, M., & Weng, T. (2010). Occupancy-Driven Energy Management for Smart Building Automation. *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building (BuildSys '10)* (pp. 1-6). Zurich: ACM.

Akyildiz, I. F., & Vuran, M. C. (2010). *Wireless Sensor Networks.* Chichester, West Sussex, UK: John Wiley & Sons Ltd.

Akyildiz, I. F., Su, W., Sankarasubrama, Y., & Cayirci, E. (2002). A Survey on Sensor Networks . *IEEE Communications Magazine , 40* (8), 102 - 114 .

Alageswaran, R., Kiruthiga, O. G., Keerthika, T. K., & Prakash, B. (2013). Design and Development of Routing Protocol for WSN Simulation in GloMoSim. *Journal of Artificial Intelligence , 6* (2), 181-186.

Anand, N., Aryafar, E., & Knightly, E. W. (2010). WARPLab: A Flexible Framework for Rapid Physical Layer Design. *Proceedings of the 2010 ACM workshop on Wireless of the students, by the students, for the students (S3 '10)* (pp. 53-56). Chicago, USA: ACM.

ANRG. (2009). *Tutornet: A Low Power Wireless IoT Testbed*. Retrieved 2015, from Tutornet: A Low Power Wireless IoT Testbed: http://anrg.usc.edu/www/tutornet/

Bagrodia, R., Meyer , R., Takai, M., Chen, Y.-a., Zeng, X., Martin, J., et al. (1998). Parsec: A Parallel Simulation Environment for Complex Systems. *Computer , 31* (10), 77-85.

Barrenetxea, G., Ingelrest, F., Schaefer, G., & Vetterli, M. (2008). The Hitchhiker's Guide to Successful Wireless Sensor Network Deployments. *Proceedings of the 6th ACM conference on Embedded network sensor systems (SenSys '08)* (pp. 43-56). Raleigh, NC, USA: ACM.

Boano, C. A., Tsiftes, N., Voigt, T., Brown, J., & Roedig, U. (2010). The Impact of Temperature on Outdoor Industrial Sensornet Applications. *IEEE Transactions on Industrial Informatics , 6* (3), 1551-3203.

Brown, S., & Sreenan, C. J. (2013). Software Updating in Wireless Sensor Networks: A Survey and Lacunae. *Journal of Sensor and Actuator Networks* , 717-760.

Castalia. (2007). Retrieved 2015, from https://castalia.forge.nicta.com.au/index.php/en/index.html

Chun, B. N., Buonadonna, P., AuYoung, A., Ng, C., Parkes, D. C., Shneidman, J., et al. (2005). Mirage: a Microeconomic Resource Allocation System for Sensornet Testbeds. *Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors (EmNets '05)* (pp. 19-28). Sydney, Australia: IEEE Computer Society.

Chaudhary, R., Sethi, S., & Keshari, R. (2012). A study of comparison of Network Simulator-3 and Network Simulator-2. *International Journal of Computer Science and Information Technologies (IJCSIT) , 3* (1), 3085-3092.

Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., & Shenker, S. (2003). Making Gnutella-like P2P Systems Scalable. *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '03)* (pp. 407-418). Karlsruhe, Germany: ACM.

Chandy, K. M., & Sherman, R. (1989). The Conditional Event Approach to Distributed Simulation. *Distributed Simulation Conference.* Miami.

Chatzigiannakis, I., Fischer, S., Koninis, C., Mylonas, G., & Pfisterer, D. (2009). WISEBED: An Open Large-Scale Wireless Sensor Network Testbed. *First International Conference on Sensor Applications, Experimentation, and Logistics (SENSAPPEAL 2009)* (pp. 68-87). Athens, Greece: Springer.

Chen, H., Tse, C. K., & Feng, J. (2009). Impact of Topology on Performance and Energy Efficiency in Wireless Sensor Networks for Source Extraction. *IEEE Transactions on Parallel and Distributed Systems , 20* (6), 886 - 897.

Chiasserini, C.-F., & Garetto, M. (2004). Modeling the Performance of Wireless Sensor Networks. *Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004).* Hong-Kong, China: IEEE Computer Society.

Coulson, G., Porter, B., Chatzigiannakis, I., Koninis, C., Fischer, S., Pfisterer, D., et al. (2012). Flexible Experimentation in Wireless Sensor Networks. *Communications of the ACM , 55* (1), 82-90.

Colesanti, U. M., Crociani, C., & Vitale, A. (2007). On the Accuracy of OMNeT++ in the Wireless Sensor Networks Domain: Simulation vs. Testbed. *4th ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN'07)* (pp. 25-31). Chania, Crete Island, Greece: ACM.

Corke, P., Wark, T., Jurdak, R., Hu, W., Valencia, P., & Moore, D. (2010). Environmental Wireless Sensor Networks. *Proceedings of the IEEE , 98* (11), 1903-1917.

Crawley, D. B., Lawrie, L. K., Winkelmann, F. C., Buhl, W. F., Huang, Y. J., Pedersen, C. O., et al. (2001). EnergyPlus: A New-Generation Building Energy Simulation. *Energy and Buildings - BUILDING SIMULATION '99 , 33* (4), 319–331.

Crossbow. (2003). *MICA2 Datasheet.* Retrieved 2015, from MICA2 Datasheet: http://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf

El-Darymli, K., & Ahmed, M. H. (2012). Wireless Sensor Network Testbeds: A Survey. In A. B. Abdullah, K. Ragab, & N. Zaman, *Wireless Sensor Networks and Energy Efficiency: Protocols, Routing, and Management* (pp. 148-205). IGI Global.

Erickson, V. L., Carreira-Perpiñán, M. Á., & Cerpa, A. (2011). OBSERVE: Occupancy-Based System for Efficient Reduction of HVAC Energy. *Proceedings of the 10th International Conference on Information Processing in Sensor Networks (IPSN 2011)* (pp. 258-269). Chicago: IEEE Computer Society.

Eriksson, J., Österlind, F., Finne, N., Tsiftes, N., Dunkels, A., Voigt, T., et al. (2009). COOJA/MSPSim: Interoperability Testing for Wireless Sensor Networks. *Proceedings of the 2nd International Conference on Simulation Tools and Techniques (Simutools '09).* Rome: ICST.

Ertin, E., Arora, A., Ramnath, R., Nesterenko, M., Naik, V., Bapat, S., et al. (2006). Kansei: A Testbed for Sensing at Scale. *The Fifth International Conference on Information Processing in Sensor Networks (IPSN 2006)* (pp. 399-406). Nashville: IEEE Computer Society.

Ettus Research. (2010). *Universal Software Radio Peripheral - The Foundation for Complete Software Radio Systems.*

Du, W., Mieyeville, F., & Navarro, D. (2011). IDEA1: A validated SystemC-based system-level design and simulation environment for wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking , 143*.

Du, W., Mieyeville, F., Navarro, D., O'Connor, I., & Carrel, L. (2014). Modeling and Simulation of Networked Low-Power Embedded Systems: A Taxonomy. *EURASIP Journal on Wireless Communications and Networking , 106*.

Du, S., Saha, A. K., & Johnson, D. B. (2007). RMAC: A Routing-Enhanced Duty-Cycle MAC Protocol for Wireless Sensor Networks. *26th IEEE International Conference on Computer Communications (INFOCOM 2007)* (pp. 1478-1486). Anchorage: IEEE Computer Science.

Dubois-Ferriere, H., Meier, R., Fabre, L., & Metrailler, P. (2006). TinyNode: A Comprehensive Platform for Wireless Sensor Network Applications. *The Fifth International Conference on Information Processing in Sensor Networks (IPSN 2006)* (pp. 358-365). Nashville: IEEE Computer Society.

Dunkels, A., Gronvall, B., & Voigt, T. (2004). Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors. *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN '04)* (pp. 455-462). Tampa, Florida: IEEE Computer Society.

Dutta, P., & Culler, D. (2008). Epic: An Open Mote Platform for Application-Driven Design. *International Conference on Information Processing in Sensor Networks (IPSN '08)* (pp. 547-548). St. Louis, Missouri, USA: IEEE Computer Society.

Des Rosiers, C. B., Chelius, G., Fleury, E., Fraboulet, A., Gallais, A., Mitton, N., et al. (2011). SensLAB Very Large Scale Open Wireless Sensor Network Testbed. *7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCOM 2011)* (pp. 239-254). Shanghai, China: Springer.

Dietrich, I., & Dressler, F. (2009). On the Lifetime of Wireless Sensor Networks. *ACM Transactions on Sensor Networks (TOSN) , 5* (1), 5.1-5.39.

Dimokas, N., Katsaros, D., & Manolopoulos, Y. (2007). Node Clustering in Wireless Sensor Networks by Considering Structural Characteristics of the Network Graph. *Fourth International Conference on Information Technology: New Generations (ITNG 2007)* (pp. 122-127). Las Vegas: IEEE Computer Society.

DOE-2. (1998). Retrieved 2015, from http://www.doe2.com/

Doddavenkatappa, M., Chan, M. C., & A. L., A. (2011). Indriya: A Low-Cost, 3D Wireless Sensor Network Testbed. *7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2011)* (pp. 302-316). Shanghai, China: Springer.

Dong, W., Chen, C., Liu, X., & Bu, J. (2010). Providing OS Support for Wireless Sensor Networks: Challenges and Approaches. *IEEE Communications Surveys & Tutorials , 12* (4), 519-530 .

Dong, W., Chen, C., Liu, X., Liu, Y., Bu, J., & Zheng, K. (2011). SenSpire OS: A Predictable, Flexible, and Efficient Operating System for Wireless Sensor Networks. *IEEE Transactions on Computers , 60* (1), 1788-1801 .

Dong, B., & Andrews, B. (2009). Sensor-based Occupancy Behavioral Pattern Recognition for Energy and Comfort Management in Intelligent Buildings. *Eleventh International IBPSA Conference, Proceedings of Building Simulation 2009.* Glasgow.

Farooq, M. O., & Kunz, T. (2011). Operating Systems for Wireless Sensor Networks: A Survey. *Sensors , 11* (6), 5900-5930.

Ferencik, I., Niemi, T., & Jolma, A. (2010). On site environmental modeling and monitoring: the Nordic Scenario in HYDROSYS project. *International Congress on Environmental Modelling and Software Modelling for Environment's Sake.* Ottawa, Canada.

Frank, C., & Römer, K. (2005). Algorithms for Generic Role Assignment in Wireless Sensor Networks. *Proceedings of the 3rd international conference on Embedded Networked Sensor Systems (SenSys '05)* (pp. 230-242). San Diego: ACM.

Guestrin, C., Bodik, P., Thibaux, R., Paskin, M., & Madden, S. (2004). Distributed Regression: an Efficient Framework for Modeling Sensor Network Data. *Proceedings of the 3rd international symposium on Information processing in sensor networks (IPSN '04)* (pp. 1-10). Berkeley, California, USA: ACM.

Gay, D., Levis, P., Behren, R. v., Welsh, M., Brewer, E., & Culler, D. (2003). The nesC Language: A Holistic Approach to Networked Embedded Systems. *Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation (PLDI '03)* (pp. 1-11). San Diego: ACM.

Ganesan, D., Estrin, D., Woo, A., Culler, D., Krishnamachari, B., & Wicker, S. (2002). *Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks.* UCLA, Computer Science Department.

Gerla, M. (2005). From Battlefields to Urban Grids: New Research Challenges in Ad Hoc Wireless Networks. *Pervasive and Mobile Computing , 1* (1), 77-93.

GDB: The GNU Project Debugger. (2006). Retrieved 2015, from https://www.gnu.org/software/gdb/

Girod, L., Stathopoulos, T., Ramanathan, N., Elson, J., Estrin, D., Osterweil, E., et al. (2004). A System for Simulation, Emulation, and Deployment of Heterogeneous Sensor Networks. *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)* (pp. 201-213). Baltimore, Maryland: ACM.

Gluhak, A., Krco, S., Nati, M., Pfisterer, D., Mitton, N., & Razafindralambo, T. (2011). A Survey on Facilities for Experimental Internet of Things Research. *IEEE Communications Magazine , 49* (11), 58-67.

Gnuplot. (1986). Retrieved 2015, from http://www.gnuplot.info/

Gracanin, D., Eltoweissy, M., Olariu, S., & Wadaa, A. (2004). On Modeling Wireless Sensor Networks. *Parallel and Distributed Processing Symposium (IPDPS 2004).* Santa Fe, New Mexico, USA: IEEE Computer Society.

Iyer, Y. G., Gandham, S., & Venkatesan, S. (2005). STCP: A Generic Transport Layer Protocol for Wireless Sensor Networks. *Proceedings of the 14th International Conference on Computer Communications and Networks (ICCCN 2005)* (pp. 449-454). San Diego, California, USA: IEEE Computer Society.

Handziski, V., Kopke, A., Willig, A., & Wolisz, A. (2006). TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Networks. *Proceedings of the 2nd International Workshop on Multi-hop Ad Hoc Networks: from Theory to Reality (REALMAN '06)* (pp. 63-70). Florence, Italy: ACM.

Hammoudeh, M., Newman, R., & Mount, S. (2008). Modelling Clustering of Wireless Sensor Networks with Synchronised Hyperedge Replacement. *4th International Conference Graph Transformations (ICGT 2008)* (pp. 490-492). Leicester, United Kingdom: Springer.

Hiranandani, D., Obraczka, K., & Garcia-Luna-Aceves, J. (2013). MANET protocol simulations considered harmful: the case for benchmarking. *IEEE Wireless Communications , 20* (4), 399-411.

Ju, X., Zhang, H., & Sakamuri, D. (2012). NetEye: A User-Centered Wireless Sensor Network Testbed for High-Fidelity, Robust Experimentation. *International Journal of Communication Systems , 25* (9), 1213-1229.

Jin, Z.-Y., & Gupta, R. (2008). Improved Distributed Simulation of Sensor Networks Based on Sensor Node Sleep Time. *4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2008)* (pp. 201-218). Santorini, Greece: Springer.

Jiménez-González, A., Martínez-de Dios, J. R., & Ollero, A. (2011). An Integrated Testbed for Cooperative Perception with Heterogeneous Mobile and Static Sensors. *Sensors , 11* (12), 11516-11543.

Jha, V., & Bagrodia, R. L. (1993). Transparent Implementation of Conservative Algorithms in Parallel Simulation Languages. *Simulation Conference Proceedings*, *Winter.*

Johnson, D., Stack, T., Fish, R., Montrallo, D., Leigh, F., Robert, S., et al. (2006). Mobile emulab: A robotic wireless and sensor network testbed. *25th IEEE International Conference on Computer Communications (INFOCOM 2006)* (pp. 1-12). Barcelona, Spain: IEEE Computer Society.

Kurkowski, S., Camp, T., & Colagrosso, M. (2005). MANET Simulation Studies: The Incredibles. *Mobile Computing and Communications Review , 9* (4), 50-61.

Keshav, S. (1988). *REAL: A Network Simulator.* Berkeley: University of California at Berkeley.

Kdouh, H., Farhat, H., Zaharia, G., Brousseau, C., Grunfelder, G., & Zein, G. E. (2012). Performance analysis of a hierarchical shipboard Wireless Sensor Network. *23rd International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2012)* (pp. 765-770). Sydney, Australia: IEEE Computer Society.

Kling, R., Adler, R., Huang, J., Hummel, V., & Nachman, L. (2004). Intel Mote: Using Bluetooth in Sensor Networks. *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys 2004)* (p. 318). Baltimore, Maryland, USA: ACM.

Koliousis, A., & Sventek, J. (2007). *Proactive vs Reactive Routing for Wireless Sensor Networks .* University of Glasgow, Department of Computing Science. Glasgow, UK: University of Glasgow.

Kotz, D., Newport, C., Gray, R. S., Liu, J., Yuan, Y., & Elliott, C. (2004). Experimental evaluation of wireless simulation assumptions. *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems (MSWiM '04)* (pp. 78-82). Venice, Italy: ACM.

Langendoen, K., Baggio, A., & Visser, O. (2006). Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture. *Proceedings of the 20th international conference on Parallel and distributed processing (IPDPS '06)* (p. 174). Rhodes Island: iEEE Computer Society.

Levis, P. A. (2006). TinyOS: An Operating System for Sensor Networks. *Proceedings of the 7th International Conference on Mobile Data Management (MDM 06)* (p. 63). IEEE Computer Society.

Levis, P., Lee, N., Welsh, M., & Culler, D. (2003). TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. *Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys '03)* (pp. 126-137). Los-Angeles: ACM.

Liu, X., Hou, K. M., de Vaulx, C., Shi, H., & Gholami, K. E. (2014). MIROS: A Hybrid Real-Time Energy-Efficient Operating System for the Resource-Constrained Wireless Sensor Nodes. *Sensors , 14* (9), 17621–17654.

Lim, C.-C., Low, Y.-H., Cai, W., Hsu, W. J., Huang, S. Y., & Turner, S. J. (1998). An Empirical Comparison of Runtime Systems for Conservative Parallel Simulation. In *Parallel and Distributed Processing* (pp. 123-134). Orlando, Florida, USA: Springer.

Lim, R., Ferrari, F., Zimmerling, M., Walser, C., Sommer, P., & Beutel, J. (2013). FlockLab: A Testbed for Distributed, Synchronized Tracing and Profiling of Wireless Embedded Systems. *International Conference on Information Processing in Sensor Networks (IPSN 2013)* (pp. 153-165). Philadelphia, USA: IEEE Computer Society.

Lo, S.-H., Ding, J.-H., Hung, S.-J., Tang, J.-W., Tsai, W.-L., & Chung, Y.-C. (2007). SEMU: A Framework of Simulation Environment for Wireless Sensor Networks with Co-simulation Model. *Second International Conference in Advances in Grid and Pervasive Computing (GPC 2007)* (pp. 672-677). Paris, France: Springer.

Nam: Network Animator. (2002, July 3). Retrieved 4 8, 2015, from http://www.isi.edu/nsnam/nam/

Nati, M., Gluhak, A., Abangar, H., & Headley, W. (2013). SmartCampus: A User-centric Testbed for Internet of Things Experimentation. *16th International Symposium on Wireless Personal Multimedia Communications (WPMC 2013)* (pp. 1-6). Atlantic City: IEEE COmputer Society.

National Institute of Building Sciences. (2015). *BLAST*. Retrieved 2015, from http://www.wbdg.org/tools/blast.php

National Science Foundation. (2002). *Report of NSF Workshop on Network Research Testbeds.* National Science Foundation, Chicago, USA.

NED. (1998). Retrieved 2015, from http://www.ewh.ieee.org/soc/es/Nov1999/18/ned.htm

NS-3. (2011). Retrieved April 7, 2015, from NS-3: https://www.nsnam.org/

Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., & Anderson, J. (2002). Wireless Sensor Networks for Habitat Monitoring. *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications (WSNA '02)* (pp. 88-97). Atlanta, Georgia, USA: ACM.

Mangharam, R., Rowe, A., & Rajkumar, R. (2007). FireFly: A Time Synchronized Real-Time Sensor Networking Platform. *Real-Time Systems , 37* (3), 183-231 .

Martin, R. C. (2003). *Agile Software Development: Principles, Patterns, and Practices.* Prentice Hall.

MathWorks. (1994). *Matlab - The language of Technical Computing*. Retrieved 2015, from http://www.mathworks.com/products/matlab/

MEMSIC. (2011). *IRIS Datasheet*. Retrieved 2015, from IRIS Datasheet: http://www.memsic.com/userfiles/files/Datasheets/WSN/6020-0124-01_B_IRIS.pdf

MEMSIC. (2004). *MICAz Datasheet.* Retrieved 2015, from MICAz Datasheet: http://www.memsic.com/userfiles/files/Datasheets/WSN/6020-0060-04-B_MICAz.pdf

MEMSIC. (2004). *TelosB Datasheet.* Retrieved 5 20, 2015, from MEMSIC Inc. - Wireless Sensor Networks: http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf

Meshkova, E., Riihijärvi, J., Oldewurtel, F., Jardak, C., & Mähönen, P. (2008). Service-Oriented Design Methodology for Wireless Sensor Networks: A View through Case Studies. *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC 2008)* (pp. 146-153). Taichung, Taiwan: IEE Computer Society.

Merrett, G. V., White, N. M., Harris, N. R., & Al-Hashimi, B. M. (2009). Energy-Aware Simulation for Wireless Sensor Networks. *Proceedings of the 6th Annual IEEE communications society conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'09)* (pp. 1-8). Rome, Italy: IEEE Computer Society.

Misra, J. (1986, March). Distributed Discrete Event Simulation. *ACM Computing Surveys* .

Mouradian, A., Augé-Blum, I., & Valois, F. (2014). RTXP : A Localized Real-Time Mac-Routing Protocol for Wireless Sensor Networks. *Computer Networks , 67*, 43–59.

MOTEIV. (2005). *Tmote Sky Datasheet.* Retrieved 2015, from Tmote Sky Datasheet:

http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf

Österlind, F., Eriksson, J., & Dunkels, A. (2010). Cooja TimeLine: A Power Visualizer for Sensor Network Simulation. *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys '10))* (pp. 385-386). Zurich: ACM.

Österlind, F., Dunkels, A., Eriksson, J., Finne, N., & Voigt, T. (2006). Cross-Level Sensor Network Simulation with COOJA. *The 31st Annual IEEE Conference on Local Computer Networks (LCN '06)* (pp. 641-648). Tampa, Florida: IEEE Computer Society.

Qualnet. (2008). Retrieved 2015, from http://web.scalable-networks.com/content/qualnet

Qutaiba, A. I. (2012). Simulation Framework of Wireless Sensor Network (WSN) Using MATLAB/SIMULINK Software. In V. Katsikis (Ed.), *MATLAB - A Fundamental Tool for Scientific Computing and Engineering Applications - Volume 2.* InTech.

Purdue University. (2008). *MAP - Purdue University Wireless Mesh Network Testbed*. Retrieved 2015, from MAP - Purdue University Wireless Mesh Network Testbed: https://engineering.purdue.edu/MESH

Papazoglou, M. (2003). Service-Oriented Computing: Concepts, Characteristics and Directions. *Proceedings of the Fourth International*

Conference on Web Information Systems Engineering (WISE 2003) (pp. 3-12). Rome, Italy: IEEE Computer Society.

Papadopoulos, G. Z., Beaudaux, J., Gallais, A., Noel, T., & Schreiner, G. (2013). Adding value to WSN simulation using the IoT-LAB experimental platform. *IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob '03)* (pp. 485-490). Lyon, France: IEEE Computer Society.

Petritsch, H. (2006). *Service-Oriented Architecture (SOA) vs. Component Based Architecture.* Technology University of Vienna, Secure Systems Lab. Vienna: TU Wien.

Salem, A. O., & Awwad, H. (2014, June). Mobile Ad-hoc Network Simulators: A Survey and Comparisons. *International Journal of P2P Network Trends and Technology (IJPTT)* .

Santini, S. (2009). tinyLAB: A Matlab-Based Framework for Interaction with Wireless Sensor Networks. *The First European TinyOS Technology Exchange (ETTX 2009).* Cork.

Sarkar, N. I., & Halim, S. A. (2011, March). A Review of Simulation of Telecommunication Networks: Simulators, Classification, Comparison, Methodologies and Recommendations. *Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT)* .

Schroth, C., & Janner, T. (2007, May 29). Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services. *IT Professional , 9* (3), pp. 36-41.

Sikka, P., Corke, P., Overs, L., Valencia, P., & Wark, T. (2007). Fleck - A Platform for Real-World Outdoor Sensor Networks. *3rd International Conference on Intelligent Sensors, Sensor Networks and Information (ISSNIP 2007)* (pp. 709-714). Melbourne, Australia: IEEE Computer Society.

Shnayder, V., Hempstead, M., Chen, B.-r., Allen, G. W., & Welsh, M. (2004). Simulating the Power Consumption of Large-Scale Sensor Network Applications. *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys '04)* (pp. 188-200). Baltimore: ACM.

Soroush, H., Banerjee, N., Corner, M. D., Levine, B. N., & Lynn, B. (2009). DOME: A Diverse Outdoor Mobile Testbed. *Proceedings of the 1st ACM International Workshop on Hot Topics of Planet-Scale Mobility Measurements (HotPlanet '09)* (pp. 1-6). Kraków, Poland: ACM.

Stecklina, O., Vater, F., Basmer, T., Bergmann, E., & Menzel, H. (2011). Hybrid Simulation Environment for rapid MSP430 system design test and validation using MSPsim and SystemC. *14th IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS 2011)* (pp. 167-170). Cottbus, Germany: IEEE Computer Society.

Stojmenovic, I. (2008, December). Simulations in wireless sensor and ad hoc networks: matching and advancing models, metrics, and solutions. pp. 102-107.

Raychaudhuri, D., Seskar, I., Ott, M., Ganu, S., Ramachandran, K., Kremo, H., et al. (2005). Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols. *IEEE Wireless Communications and Networking Conference (WCNC 2005). 3*, pp. 1664-1669. New Orleans, Louisiana, USA: IEEE Computer Society.

Rensfelt, O., Hermans, F., Gunningberg, P., Larzon, L.-Å., & Björnemo, E. (2011). Repeatable Experiments with Mobile Nodes in a Relocatable WSN Testbed. *The Computer Journal , 54* (12), 1973-1986.

Ricci, R., Duerig, J., Stoller, L., Wong, G., Chikkulapelly, S., & Seok, W. (2012). Designing a Federated Testbed as a Distributed System. *8th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2012)* (pp. 321-337). Thessaloniki, Greece: Springer.

Rice University. (2006). *WARP Project*. Retrieved 2015, from WARP Project: http://warpproject.org/trac/wiki/about

Riley, G. F. (2012). Retrieved 2015, from NetAnim: https://www.nsnam.org/wiki/NetAnim

Riliskis, L., & Osipov, E. (2015). Symphony : A Framework for Accurate and Holistic WSN Simulation. *Sensors , 15* (3), 4677-4699.

Ringwald, M., & Romer, K. (2007). Deployment of Sensor Networks: Problems and Passive Inspection. *Fifth Workshop on Intelligent Solutions in Embedded Systems* (pp. 179-192). Leganes, Spain: IEEE Computer Society.

Roth, A. (2013). *EnergyPlus Boosts Building Efficiency with Help from Autodesk*. Retrieved 2015, from Department of Energy: http://energy.gov/eere/articles/energyplus-boosts-building-efficiency-help-autodesk

Titzer, B., Lee, D. K., & Palsberg, J. (2005). Avrora: Scalable Sensor Network Simulation with Precise Timing. *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks (IPSN '05)* (pp. 477-482). Los Angeles: IEEE Computer Society.

The Network Simulator - NS-2. (2011, November 5). Retrieved April 8, 2015, from http://nsnam.isi.edu/nsnam/index.php/User_Information

Tonneau, A.-S., Mitton, N., & Vandaele, J. (2014). A Survey on (mobile) Wireless Sensor Network Experimentation Testbeds. *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)* (pp. 263-268). Marina Del Rey, California, USA: IEEE Compter Society.

List of Figures

List of Tables

Abbreviations

ACK	Acknowledgment
ADC	Analog-to-Digital Converter
ANP	Accelerated Null Message Protocol
AP	Access Point
API	Application Programming Interfaces
ARQ	Automatic Repeat Request
BB	Back-Bone
BC	Back-Channel
CAN	Campus Area Network
CPU	Central Processing Unit
CSMA	Carrier Sense Multiple Access
CT	Cluster Testbed
DTN	Delay Tolerant Network
EPROM	Erasable Programmable Read-Only Memory
FEC	Forward Error Correction
FPGA	Field-Programmable Gate Array
FT	Federated Testbed
GUI	Graphical User Interface
GW	Gateway
IoT	Internet-of-Things
ISS	Instruction Set Simulator
IT	Information Technology
JNI	Java Native Interface
LAN	Local Area Network
MAC	Medium Access Control
MAN	Metropolitan Area Network
MANET	Mobile Ad Hoc Network
MXT	Multi-User Experimental Testbeds
NENM	Node Emulators with Network Models

NSF	National Science Foundation
NSLU	Network Link Storage Units
NSNE	Network Simulators with Node Emulators
NSNM	Network Simulators with Node Models
NSSNM	Node System Simulator with Network Models
OS	Operating System
OSI	Open Systems Interconnection
OT	Overlay Testbed
P2P	Peer-to-Peer
PAN	Personal Area Network
Pcap	Packet Capture
PCT	Proof-of-Concept Testbeds
PDA	Personal Digital Assistant
PH	Processing Hub
QoS	Quality-of-Service
RAM	Radom Access Memory
RF	Radio Frequency
RK	Research Kit
S/CAN	System-Cluster Area Network
SAN	Storage Area Network
SEB	Standard Extension Board
SH	Storage Hub
SLDL	System-Level Description Languages
SOA	Service-Oriented Architecture
SSH	Secure Shell
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USRP	Universal Software Radio Peripheral
WAN	Wide Area Network
WARP	Wireless Open-Access Research Platform
WLAN	Wireless Local Area Network
WMN	Wireless Mesh Network
WSN	Wireless Sensor Network