One Step Forward: Linking Wireless Self-Organizing Network Validation Techniques with Formal Testing Approaches

ALINE CARNEIRO VIANA, INRIA
STEPHANE MAAG, TELECOM & Management SudParis and Samovar CNRS
FATIHA ZAIDI, Université Paris-Sud, LRI UMR 8623

Wireless self-organizing networks (WSONs) have attracted considerable attention from the network research community; however, the key for their success is the rigorous validation of the properties of the network protocols. Applications of risk or those demanding precision (like alert-based systems) require a rigorous and reliable validation of deployed network protocols. While the main goal is to ensure the reliability of the protocols, validation techniques also allow the establishment of their correctness regarding the related protocols' requirements. Nevertheless, even if different communities have carried out intensive research activities on the validation domain, WSONs still raise new issues for and challenging constraints to these communities. We thus, advocate the use of complementary techniques coming from different research communities to efficiently address the validation of WSON protocols. The goal of this tutorial is to present a comprehensive review of the literature on protocol engineering techniques and to discuss difficulties imposed by the characteristics of WSONs on the protocol engineering community. Following the formal and nonformal classification of techniques, we provide a discussion about components and similarities of existing protocol validation approaches. We also investigate how to take advantage of such similarities to obtain complementary techniques and outline new challenges.

Categories and Subject Descriptors: C.2.2 [**Network Protocols**]: *Protocol verification*; C.4 [**Performance of Systems**]: *Modeling techniques*; I.6.0 [**Simulation and Modeling**]: General

General Terms: Algorithms, Verification, Performance

Additional Key Words and Phrases: Protocol validation, wireless self-organizing networks

ACM Reference Format:
Viana, A. C., Maag, S., and Zaidi, F. 2011. One step forward: Linking wireless self-organizing network validation techniques with formal testing approaches. ACM Comput. Surv. 43, 2, Article 7 (January 2011), 36 pages.
DOI = 10.1145/1883612.1883614 http://doi.acm.org/10.1145/1883612.1883614

1. INTRODUCTION

1.1. Context

"It is not the biggest nor the fastest of the species that survive, but the one that adapts to its environment." (Charles Darwin, *Theory of Evolution*, 1809–1882).

Nature is full of interesting examples of systems with self- (self-configuration, self-organization, etc.) properties, constituting a valuable source of inspiration for the engineering of fully autonomous formation of networks. In addition, advances in com-

Authors' addresses: A. C. Viana, INRIA Saclay-Ile de France sud, Parc Club Orsay, Université ZAC des vignes, 4, rue Jacques Monod, 91893 Orsay Cedex, France; email: aline.viana@inria.fr; S. Maag, TELECOM SudParis, CNRS UMR 5157-Samovar, 9, rue Charles Fourier, 91011 EVRY Cedex, France; email: stephane.maag@itsudparis.eu; F. Zaidi, Université Paris-Sud, LRI, bât 490 Orsay, F-91405, France; CNRS, Orsay, F-91405, France; email: fatiha.zaidi@lri.fr.

munication technologies and the proliferation of wireless computing and communication devices are opening new ways for mobile users to get connected to each other. As a consequence, autonomic networks have emerged with the goal of relying on processes of evolution, development, self-organization, adaptation, learning, teaching, and goal orientation. This futurist goal can be represented by the design of multihop wireless self-organizing networks (WSONs) that are able to robustly respond to dynamically changing environments, operating conditions, and purposes or practices of use, thus facilitating new ways to perform network control, management, and service creation. Wireless networks such as sensor networks, mesh networks, vehicular networks, delay-tolerant networks, and MANETs are some examples of networks that follow the principle of WSONs.

Over the last number of years, multihop wireless networking area has thus attracted considerable attention within both industry and academia. One reason for this popularity is the wide range of novel applications in the areas of health, military, environment, and home. The requirements of such applications have, however, a direct impact on the design of the wireless network. In military areas, for instance, rapid deployment, self-organization, and fault tolerance characteristics are required. In environmental areas, reliability, fault tolerance, and robustness are important issues, and constitute fundamental characteristics, for instance, in alert-based monitoring applications.

Hence, it can be easily concluded that the success/quality of those applications is then strongly related to the correctness and good performance of the involved network protocols. In particular, safety-critical applications (like healthcare-related or alert-based systems) require a rigorous and reliable validation of all network functionalities and features.[1] In addition to threatening people's lives, faulty software has a financial impact. The fact that people rely on computers in practically every aspect of their lives (e.g., in cars, ATMs, cell phones, etc.) makes higher the cost of unreliable design [Hoffman 2008].

1.2. Motivation

Many efforts have been performed in order to validate the requirements and the functioning of protocols in such kinds of networks. While the main goal is to ensure the reliability of the protocols, validation techniques also allow the establishment of their correctness with respect to the related requirements. In particular, the properties to be validated are related to behavioral aspects, which are commonly known as *functional* or *qualitative* (e.g., protocol interactions, or loop free) and *nonfunctional* or *quantitative* properties (e.g., performance-related issues, like latency, delivery ratio, etc.). In this way, validation techniques have been studied by the research community through different approaches. In particular, functional and/or nonfunctional properties have been validated by the use of *formal* or *nonformal* approaches.

In the multihop wireless networking area, the major techniques used to design and ensure the quality of the network-related protocols essentially rely on descriptions for simulation and/or emulations, even if some works put also trust in mathematical models for the understanding of systems' behavior. More specifically, the majority of works rely on nonformal models provided as input to simulators such as NS-2,[2] OPNET,[3] or GloMoSim.[4] In this case, simulation is usually conceived to observe and analyze the protocol performance. Nevertheless, works in the literature [Cavin et al. 2002; Kurkowski et al. 2005; Andel and Yasinsac 2006] have shown that there are

[1]Here reliability means that all the application's behaviors are correct against all specified criteria.
[2]Network Simulator `www.isi.edu/nsnam/ns`.
[3]OPNET Simulator. `www.opnet.com`.
[4]Global Mobile Information Systems Simulator. `pcl.cs.ucla.edu/projects/glomosim/`.

growing concerns regarding the reliability of results generated by wireless network simulators. In addition, they have also discussed the scarcity of results gotten from real experiments [Abdesslem et al. 2007] and the huge diversity of results from simulation when compared to the ones from real case studies. Otherwise, even if emulation testing (TinyOS Simulation, or TOSSIM[5]; WSim[6]) comes closer to the reality, the simulation test is still required and represents an important component in the emulation testing. Hence, the combination of simulation and emulation techniques is not still enough to replace a real case study [Bhargavan et al. 2002a]. Finally, although useful for evaluating nonfunctional properties of protocols, such techniques do not allow one to discern design errors or define automation of well-defined processes, important issues for evaluating the functional behaviors of protocols.

Recently, some works in the literature have then advocated the use of formal models to test WSON routing protocols [Engler and Musuvathi 2004; Fernández-Iglesias et al. 2005; Fehnker et al. 2007], as a way to deal with the previously described constraints of nonformal models. *Verification* and *testing* are two complementary stepwise techniques for formal protocol validation. The verification technique consists in a formal modeling of the protocol in order to verify some of its properties.[7] Otherwise, testing techniques work on implementations rather than models. In this way, test sequences generated from the formal model are injected into the final implementation of the protocol. This will allow the comparison between the real results and the expected results provided by the specification.

Nevertheless, formal description techniques and their testing tools have not frequently been applied in the multihop wireless networking area. This is mainly due to the difficulties that characteristics of WSONs impose to the formal modeling [Wibling 2005; Fernández-Iglesias et al. 2005]. In particular, as discussed later in this article, WSONs present a number of characteristics that set them apart from traditional wired networks, as the network dynamicity or the inherently broadcast communication. Thus, even if different communities have carried out intensive research activities on the validation domain, WSONs still raise new issues and challenging constraints to these communities. One example is the scalability issue. In the validation-related works concerning WSONs, the considered network size remains small (e.g., five nodes in de Renesse and Aghvami [2004], or 18 nodes in Fernández-Iglesias et al. [2005]). This is due to the dynamicity imposed by WSONs, which highly increases the number of states to be considered in the validation process.

1.3. Contribution

According to the literature and similarly to researches on the validation area of wired networks, it has been well established that the validation of WSONs protocols may not be addressed by only one method, i.e., formal or nonformal [Fehnker et al. 2007]. This specifically suggests the integration of formal and nonformal approaches, which still constitutes an open issue in the protocol validation domain.

Following this assumption, it can be observed that similarities can be established between formal and nonformal approaches in terms of techniques and properties to be validated. In this way, we argue that the use of complementary techniques coming from different research communities can help to efficiently address the new constraints imposed by WSONs.

[5]`www.cs.berkeley.edu/pal/research/tossim.html`.

[6]WSim Simulator. `worldsens.citi.insa-lyon.fr/joomla/index.php`.

[7]In routing protocols, formal models may check the loop free property of established routes or still the rapid convergence of routes changes.

The goal of this survey is to present a comprehensive review of the literature on protocol engineering techniques and to discuss the challenges imposed by WSONs to the protocol engineering community. Our aim is to provide a better understanding of the current research issues in this field. Following the formal and nonformal classification of techniques, we provide an overview of protocol validation approaches, investigate their design features and constraints, and discuss their similarities. We also investigate how to take advantage of such similarities to obtain complementary techniques.

The scope of the work presented in this article is distinguished in many aspects from existing surveys on validation techniques [Curren 2005; Holzmann 1991]. In particular, our work differs from other surveys as follows.

—In general, the literature presents a collection of validation-related works that are adapted to a particular type of problem (e.g., leader election, or a specific routing protocol). Instead, our goal is to help the reader understanding the *foundations* of validation techniques. In this way, we survey both formal and nonformal validation approaches and discuss their limitations and drawbacks. We then examine the attempts of convergence addressed in the literature.
—The related surveys in the literature are devoted to wired networks only. Due to the importance of WSON and its new challenging characteristics, a detailed discussion on the changes introduced to the protocol engineering domain becomes necessary and useful at this stage. Thus our work is also a dedicated study of the particularities introduced by multihop wireless networks. In addition, we investigate how validation techniques have been rethought to allow the application to WSONs.
—Finally, we discuss open research problems. We believe the provision of more general insights across the validation techniques is an interesting direction through the design of novel validation techniques adapted for WSONs.

1.4. Outline

The remainder of this survey is organized as follows. We start our analysis by providing in Section 2 an overview of the protocol engineering domain. In Section 3, we discuss the new challenges introduced by the wireless self-organizing networks that impose limitations to existing validation techniques. In Sections 4 and 5, we examine in detail the formal and nonformal approaches used in the protocol engineering domain. We then survey concepts and discuss the methods used in each approach. Section 6 provides a discussion about the advantages and drawbacks of those approaches regarding WSON validation. Section 7 examines the demands for convergence between formal and nonformal domains and discusses the interest in this convergence, in dealing with the characteristics of WSONs. Finally, Section 8 summarizes our investigations and presents our conclusions.

2. PROTOCOL ENGINEERING

The protocol engineering domain covers a large range of activities, from the requirement specification to the final deployment, passing through the design phase. There exists in the literature several way to perform the development of a protocol. All these processes of development share a common starting point.

The design and the validation of a protocol are preceded by the specification of its rules and format of messages. Messages allow the establishment of communication between entities in a distributed system. In this way, once primitives, data units, and communication rules are defined, the *protocol's lifecycle* can be then started. This consists in the execution of three main phases: the *development*, the *exploitation* (i.e. final deployment), and the *maintenance* (see Figure 1). This article addresses the development phase, important to ensure the correct design of a protocol to be deployed and

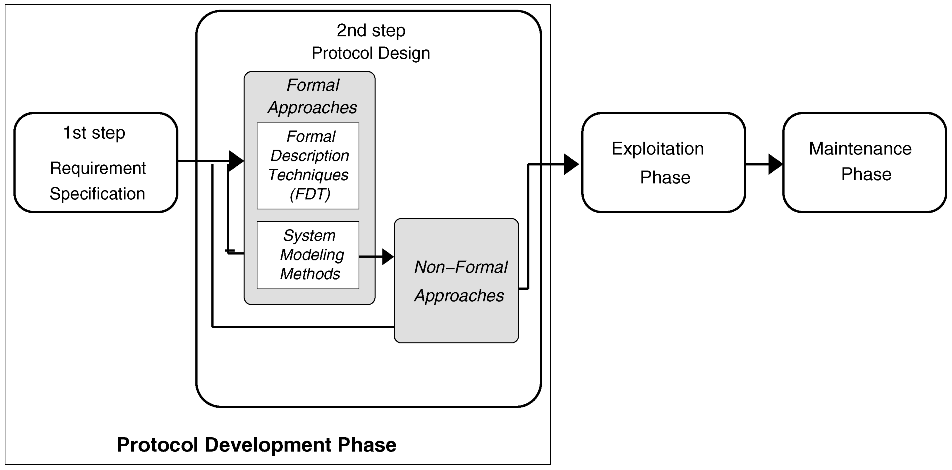

Fig. 1. Protocol lifecycle.

maintained in a real environment. The development phase can be proceed according to several models: the Waterfall [Royce 1987], V-model [Vliet 1999], spiral [Boehms 1988], prototyping, RAD [Martin 1990], and RUP [Jacobson et al. 1999] models. More specifically, protocol lifecycles can be divided into two mains classes: the linear and the iterative lifecycles. The Waterfall and the V-model are known as *linear lifecycles*. The name *Waterfall* [Royce 1987] derives from the cascading effect existent between the steps. When this model is used, no intermediate evaluation is performed between the starting point of the project and the validation step. Hence, this lack of intermediate evaluation has as consequence the fact that no way to follow the development process exists and then no means to organize a work within a team is available. Such a model increases the risk of having errors in the system due to the late validation step. Although being the least flexible, the Waterfall model is well suited for projects with few participants and when the risks are well determined from the starting point of the project.

The second linear model, the V-model [Vliet 1999], where formal methods are used, is the most famous model, being often used in big projects. For these reasons, we selected the V-model as the model of the development phase of a protocol lifecycle to be detailed in the following section. The main advantage of such a model relies on a model driven by documentation which is produced at every step of the protocol lifecycle. The risks of building a wrong system are also reduced as a validation is performed at each step. On the other hand, the weaknesses of this model may be also related to the documentation if all the attention is only concentrated on the production of this one without thinking of the main objective, that is, the protocol development. The intermediate validations cannot prevent the transmission of deficiencies caused by the previous steps of the lifecycle.

In the family of iterative models, we can cite the prototyping and spiral models [Boehms 1988]. The former model is a cyclic version of the linear model. In this model, once the requirement analysis is done and the design for a prototype is made, the development process gets started. Once the prototype is created, it is given to the customer for evaluation. The customer's response creates the next level of requirements and defines the next iteration. The main strengths are that the users can see steady progress, the feasibility of a proposed design approach can be examined, and system performance issues can be then explored. The drawbacks are caused by the possibility for the users to treat the prototype as the solution and because of the fact that a prototype is only a partial specification. Moreover, with such a model, there is no way to know the number of iterations that will be required. On the other hand, the spiral mode consists of a risk reduction by breaking a system project into miniprojects, each one addressing one or more major risks. After major risks have been addressed, the spiral model ends up as a Waterfall model. The strengths are on the early iterations, which are the cheapest and enable the higher risks to be addressed at the lowest total cost. In this way, the iterations can be tailored in an efficient way to fit the project's needs. The weaknesses have to do with the ability to make it in practice as it requires attentive management knowledge.

Other models exist such as the RAD model (Rapid Application Development) [Martin 1990], which is a linear sequential model that emphasizes an extremely short development cycle that relies on a component-based construction, and the RUP model (Rational Unified Process) [Jacobson et al. 1999], which is related to the spiral model and has an underlying object-oriented model. We can also mention the agile methodologies, which are based on an iterative process. Interested readers can refer to Jeffries and Ambler [2002].

To conclude, no lifecycle model is perfect and the choice of the right model is really dependent on the system to be developed and on the tradeoffs to be found between all the parameters considered in the system development phase, such as size project, development team size, etc. (see Vliet [1999], Chapter 3, for a good overview of lifecycle models). As stated before, we describe in the following the V-model in detail, where the development is performed in several steps, having at each step a validation process and an associated technical report. These steps are shared by several lifecycle models.

2.1. Protocol Development Phase

A protocol development cycle is divided into several steps, which are detailed in this section. The first step captures the requirements of the user or application in terms of available and requested services. This constitutes a high-level task, since no specification of how the system internally works is required. Instead, this first step specifies how the system reacts to interactions coming from the environment. Traditionally and especially in the formal-related area, the requirements are expressed by sequence diagrams, called *message sequence charts* [ITU-T 1996], which represent the exchanges of interactions between the different entities of the system and its environment.

The second step designs the protocol in terms of data structures and data units exchanged to manage the timing constraints (i.e., the management of timers). More specifically, the design of a protocol relies on the available service to supply the requested service. The protocol design can be performed by means of formal approaches, as system modeling methods or formal description techniques (FDT), or directly by nonformal description techniques.

An FDT builds a formal model of the protocol that can be used for several purposes, such as formal verification or formal validation of users' requirements, as well as to generate the tests to be executed on the real implementation. The use of an FDT allows checking the correctness of the protocol regarding its expected behavior (see Section 5).

Verification and validation (or testing) are complementary techniques for formal protocol description. At the verification step, properties related to the service and those related to the protocol can be verified. If the service-related properties are verified on the model, it can be then established that the model is correct regarding what the system is expected to do, for example, if a route between two nodes can be correctly established. Protocol-related properties are verified on the model in order to

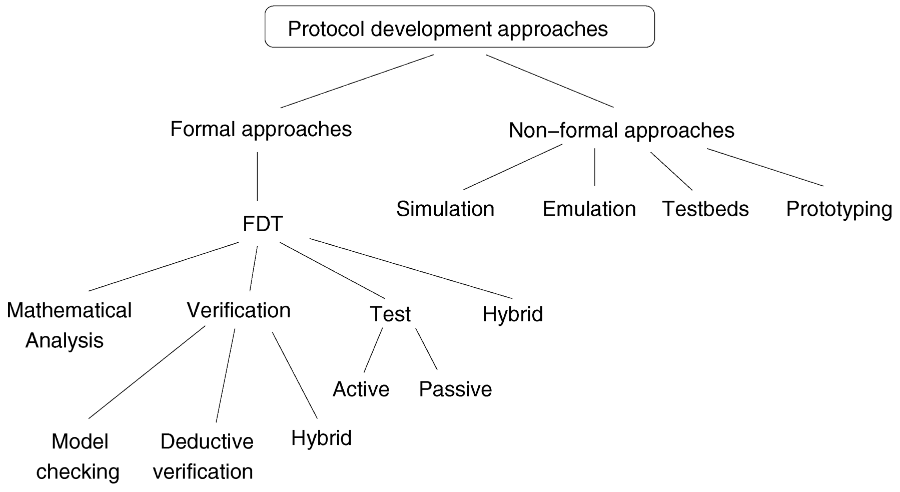

Fig. 2. General classification of protocol design approaches.

establish that the model is free of deadlocks, livelocks, etc. After the protocol verification is concluded (i.e., it is correct and corresponds to the specified requirements) then, from the correct model, test cases that cover test objectives can be automatically generated.

Finally, the V-model cycle of a protocol is finished by its implementation. If a formal description was adopted in the previous steps, the implementation can be directly generated from the formal model. There exist many code generators from formal models; however, the code generated is not complete and needs to be finalized. The tests automatically generated from the formal model can then be exercised on the final implementation to check whether the implementation conforms to its specification.

Performance analyzers are actuated after the design phase. With formal approaches (such as the system modeling methods described in Section 5.1) or with nonformal approaches (such as simulation, emulation, real-live testing, and prototyping, described in Section 4), the issue to be addressed is the performance evaluation of the developed protocol. In the realm of performance evaluation, the previous steps of requirement and design specifications are also important steps to be performed. Nevertheless, the requirements are expressed by means of scenarios directly produced from the informal description of the protocol. In the case of some nonformal approaches, for example, these scenarios are scripts to be executed on the simulated or emulated protocol. The protocol is then implemented inside the simulator/emulator according to the data structures and algorithms established at the design phase.

In summary, the design phase of the protocol engineering domain is addressed in this article by two different approaches: the formal and the nonformal. Approaches based on well-founded semantics, such as methods based on formal description techniques and on system modeling methods, are studied in this article as formal approaches. Otherwise, techniques like simulation, emulation, real world testing, and prototyping are studied as nonformal approaches. Formal and nonformal approaches are complementary techniques to be used in the development of new protocols.

Figure 2 presents the general taxonomy of the protocol development approaches. A survey of nonformal and formal approaches as well as a discussion of their strengths and weaknesses is provided in the following sections. Moreover, in order to better investigate their characteristics when compared to their use in the wired network area, the next section first describes the new challenges imposed by WSONs that affect the way protocols should be designed.

3. THE WSONS CHALLENGES

Wireless self-organizing networks introduce new challenges to the project engineering research community. This is due the fact that they present radically different technical characteristics that set them apart from wired networks. In the following, we describe the particularities of these networks. Sections 6 and 7 then discuss the difficulties imposed by the new challenges of WSONs and how validation techniques have been rethought to be applied in WSONs.

—*Broadcast communication.* Unlike in wired network, where the point-to-point model of communication is dominant, communication in wireless networks is inherently broadcast based. That is, when a node transmits some information, typically all nodes within its transmission range can receive it. The broadcast nature of wireless communication can be exploited in a number of applications, including information dissemination.

—*Communication paths determined by physical location.* Due to the wireless link properties of WSONs, neighborhood is imposed by the physically location of nodes. Nodes should then rely on their physically immediate neighbors for communication.

—*Links are unreliable.* Variables such as obstructions, interference, environmental factors, and mobility make determining connectivity a priori difficult in WSONs. Also, contrary to wired networks, in which the channel can be characterized reasonably well, much more unpredictability is expected in the wireless case. Thus, the low link reliability and its possible asymmetry require that protocol designed for WSON be fault tolerant and adaptable to connectivity changes.

—*Collision*. Contrary to wired networks, collision detection is not feasible in WSONs and collision avoidance is hard to achieve. Thus links may suffer a much higher percentage of message loss through collisions.

—*High heterogeneity*. In WSONs, it is likely that nodes are heterogeneous in their characteristics such as memory availability, computation capacity, and transmitting power.

—*Potentially high mobility*. Due to the absence of wiring, nodes in WSONs can be mobile. It has been shown in the literature that node mobility is favorable to the spread and/or aggregation of information through the network [Grossglauser and Tse 2002; Benbadis et al. 2005; Vahdat and Becker 2000]. On the other hand, node mobility also imposes changes on the topology. This dynamic topology, in turn, implies more complex management algorithms for topology maintenance and routing. For instance, the dynamic nature causes routes to be unstable and make routing a resource-greedy operation. In this way, in order to have flexibility in route selection and simpler dynamic-network management, the designed addressing structure and forwarding mechanisms should be as flexible as possible.

—*Constrained resources*. Due to the absence of wiring and their small physical size, wireless devices often have limitations in memory, processing, and above all power. In this case, the optimization of resources is strongly required in order to minimize energy consumption and communication overhead. This also requires (1) taking local-scoped decisions through simple neighborhood consensus, and (2) the distribution of information and management responsibilities among the nodes in the network.

—*Vulnerability*. Due to their infrastructure-free and nonauthority capabilities, WSONs are inherently insecure. In addition, transmissions are generally in broadcast mode, which makes traffic overhearing easier for any node.

4. NONFORMAL APPROACHES

In the network community, the major techniques used to debug and evaluate designed network protocols rely on nonformal approaches such as simulation and emulation tools, as well as live-testing experiments (real testbeds) and rapid prototyping. In the wireless network domain in particular, the advantages in hardware development have made possible the deployment of inexpensive, autonomous, and compact sensor devices, which has improved the viability of deploying live-testing experiments of wireless sensor networks [Xu et al. 2004].

Otherwise, the choice of a simulator or an emulator is up to the designer. If having a high view of one idea is enough to evaluate its performance (e.g., reliability of routing protocol, zone coverage guarantees, etc.), the simulation will be the more useful tool. If, however, a fine-tuning precision of low level-results is required (e.g., precise timing analysis of the simulated software, etc.), then the emulation will be the more effective tool.

It the following, we discuss each of these widely employed nonformal approaches. Notice, however, that our goal here is to provide discussion about the concept and characteristics of nonformal approaches, instead of surveying techniques existing in the literature for each approach. Although the discussions will be focused on sensor networks, a good survey of simulation tools can be found in Curren [2005].

4.1. Simulation

Simulation plays a valuable role in network research, allowing designers to test networking protocols in a controlled and reproducible environment. Simulations are often used as an adjunct to, or substitution for, modeling systems for which simple closed-form analytic solutions are not possible. In the wireless network domain, simulation constitutes an important tool, since the evaluation in a real environment of wireless applications requires nonnegligible programming skills and time, besides imposing limitations on the network size.[8] Researchers generally use simulation to analyze system performance (i.e., quantitative analysis) prior to physical design, or to compare multiple alternatives over a wide range of conditions.

Simulation can be defined as “the representation of the behavior or characteristics of one system through the use of another system, esp. a computer program designed for the purpose” (according to `dictionary.com` Web site). For instance, computer programs can simulate weather conditions, chemical and atomic reactions, etc. Theoretically speaking, a computer simulation of a phenomenon becomes possible if mathematical data and equations can be designed to represent it. Nevertheless, the fact that most natural phenomena are subject to an almost infinite number of influences makes their simulation an extremely difficult task in practice. Therefore, simulations are usually performed by implementing only the most important factors of a phenomenon.

Simulations are also used to test new theories, and in the context of communication and computer network research, new protocols. In the case of network protocols, after creating a theory of causal relationships or a description of interactions between the different network entities (hosts/routers, data links, packets, etc.), a network engineer can codify these interactions in the form of a computer program. This program or simulator models the behavior of a network either by calculating the interactions between the different network entities using purely mathematical models, or by capturing and playing back observations from the behavior of these network entities. If the simulated program behaves in the same way as the real case, there is a good chance that the proposed relationships/descriptions of interactions are correct.

[8]Wireless scenarios may be particularly difficult or expensive to emulate using real hardware.

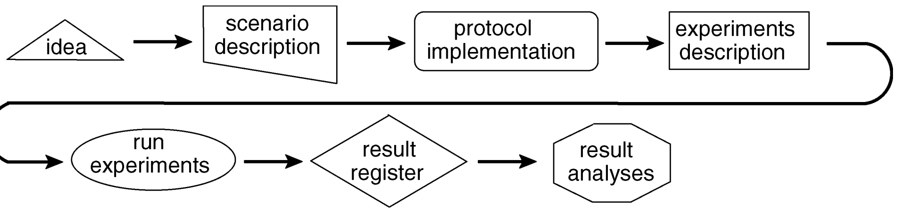

Fig. 3. Workflow describing the steps related to a protocol design based on the simulation non-formal approach.

A workflow describing the steps required in the protocol design process using the simulation nonformal approach is shown in Figure 3. After the protocol description (represented by *idea* in the workflow), the network scenario description must be provided. Considering the case of wireless network simulators, the "scenario description" specifies the topology (number of nodes, node distribution, network area size), the connectivity (the communication range, the neighborhood density), the network dynamicity (static, nodes join/leave, link failures, mobility model—individual or group mobility), and the traffic characteristics (messages types and format, traffic type, communication model—broadcast, unicast, gossip, etc.). The "implementation" of the protocol specifications is then followed by the experiment description. This latter step has as its goal to describe the metrics (delay, delivery ratio, latency, etc.) to be evaluated and consequently results in the implementation of statistic monitors. These monitors are entities that check the performance metrics during the simulation execution and generate the trace files accordingly. Trace files are generated according to the designer's needs and register monitored statistics, which will be used for final protocol analysis. Important examinations should be performed in order to guarantee the correct protocol implementation and scenario specifications [Kurkowski et al. 2005].

Typical examples of currently used networking simulators are NS-2 (see footnote 2), OPNET (see footnote 3), GloMoSim (see footnote 4), SimReal,[9] SENSE,[10] TOSSIM (see footnote 5), WSNet,[11] SensorSim,[12] J-Sim,[13] SENS,[14] EmStar [Girod et al. 2004a, 2004b],[15] and REAL [Keshav 1988]. These simulators target a higher range of protocols and provide a simulation language with network protocol libraries. Instead, the "do-it-yourself" class of simulators are much more focused implementations that usually model only the details relevant to the developer.

Most network simulators (Network Simulator; OPNET simulator; GloMoSim; WSNet; TinyOS Simulator) use discrete event simulation, in which a list of pending "events" is stored, and those events are processed in order, with some events triggering future events – such as the event of the depart of a packet at one node triggering the event of the arrival of that packet at a downstream node. Some others are classified as application-oriented simulators (SENS; EmStar) and provide a framework for developing applications on wireless sensor networks. Most systems have also an improved programming environment with a graphical user interface (GUI), while some network simulators require input scripts or scenario descriptions (network parameters: node placement, connectivity, link failures, etc.). In particular, in the NS-2 simulator,

[9]SimReal Network Simulator. `ns.l10.csie.nctu.edu.tw/`.
[10]Sensor Network Simulator and Emulator. `www.ita.cs.rpi.edu/sense/index.html`.
[11]Wireless Sensor Network Simulator. `worldsens.citi.insa-lyon.fr/joomla/index.php`.
[12]A Simulation Framework for Sensor Networks. `nesl.ee.ucla.edu/projects/sensorsim/`.
[13]Java-Sim. `www.j-sim.org/`.
[14]A Sensor, Environment and Network Simulator, `osl.cs.uiuc.edu/sens/`.
[15]Software for Wireless Sensor Networks. `cvs.cens.ucla.edu/emstar/`.

a split-level programming model is provided in which packet processing is done in the C++ system language while the simulation setup is done in a scripting language (i.e., Tcl/Tk, oTcl). A common output of simulations is the trace files.

Simulators typically come with support for the most popular protocols in use today, such as IPv4, IPv6, UDP, and TCP. Some of them, as in the cases of Networks Simulator, OPNET simulator, GloMoSim, Java-Sim, and REAL, bring the support for simulating different kind of WSONs (like wireless sensor networks, vehicular networks, MANET, etc.). On the other hand simulators like SENS, EmStar, SENSE, SensorSim, WSNet, and TinyOS Simulator were specifically designed for the wireless sensor networks' simulation.

Some works in the literature contain interesting discussions about the limitations and drawbacks of some simulators [Cavin et al. 2002; Kurkowski et al. 2005; Andel and Yasinsac 2006; Kiess and Mauvea 2007]. A survey on simulators is also contained in Curren [2005]. That survey presents characteristics of some main network simulators, discusses their advantages and disadvantages, and presents their differences.

In spite of the drawbacks described in Section 6, simulation is still considered the most widely used methodology for evaluating network protocols. This is mainly due to its scalability and reproducibility characteristics, besides allowing protocol designers to obtain some further observations that cannot be captured by analytical models. Therefore, some works in the literature discuss the pitfalls to avoid when performing simulation-based studies [Kurkowski et al. 2005; Balci 1997; Bagrodia and Takai 1999]. Pitfall examples include starting implementation without previously validating the model to be implemented; not correctly configuring all the variables of the simulator and/or leaving key parameters undefined; not properly setting the seed of the pseudorandom number generator, etc. Avoiding pitfalls can aid researchers in conducting and reporting credible simulation results.

4.2. Emulation

An emulator is different from a simulator in the way it runs actual application code. More specifically, it refers to the ability of a computer program or hardware environment to closely reproduce the features and behavior of real world devices. In this way, emulation focuses on recreating the original device environment and can be time consuming and difficult, but valuable because of its ability to maintain a closer connection to the authenticity of the real device.

In general, in the network research domain, emulators use a simulation program in conjunction with an emulated hardware in order to observe end-to-end performance of the emulated device. An emulator may thus trick a running software into believing that an emulated device is really a real device. As an example, software programs can emulate microprocessors and sensor devices. This focus on exact reproduction of external behavior contrasts with simulators, which use an abstract model of the system.

A workflow describing the steps required in a protocol emulation process is similar to the one shown in Figure 3 for simulation. The differences are basically at the level of detail required at the implementation step and the kind of generated results. Simulation tools, for example, result in statistically evaluated protocols and algorithms, while emulation tools result in tested implementations.

Some examples of wireless sensor network emulators are TOSSIM, EmStar, and WSim. Although being described as a discrete-event simulator for TinyOS applications on MICA Motes, TOSSIM (see footnote 5) is more a TinyOS emulator than a general WSON simulator. This is due to the fact that programs developed in TOSSIM can be directly targeted to Motes without modification, facilitating then the source-level application and OS debugging. EmStar (see footnote 15) is a Linux-based software framework for developing and deploying wireless sensor networks, and can be used to develop software for Mica2 motes and iPAQ-based microservers. Like WSim and TOSSIM, EmStar uses the half-simulation/half-emulation approach. WSim (see footnote 6) simulates the hardware based on the MSP430 microcontroller (MCU) series from Texas Instruments. One of the main WSim feature is its interface with the WS-Net simulator (see footnote 11) to perform the simulation of a complete sensor network. More examples of emulators are the MNE [Macker et al. 2003], NetKit [Pizzonia and Rimondini 2008],[16] and NEMAN [Puzar and Plagemann 2005]. Other emulators are listed in Rimondini [2007].

In spite of the drawbacks described in Section 6, the real advantage of emulation is the fact it allows much higher flexibility in carrying out network tests. The software piece of an emulator allows every aspect of the network be influenced and monitored like it could be in a real network, which ensures very high accuracy.

A taxonomy of a number of emulators and a discussion about the differences of each one is provided in Rimondini [2007]. Different approaches to model mobility in emulation and real world testbeds, and an overview of different evaluation techniques are provided in Kropff et al. [2006] and Kiess and Mauvea [2007].

4.3. Testbeds

In addition to theory, simulators, and emulators, there is a strong need for large-scale testbeds where real-life experimental conditions hold.

In particular, the importance of testbed-based evaluation of network protocols/applications is gaining wider recognition in the networking research community, especially in the wireless and mobility areas. This is due to the fact that testbedding real systems allows us to confront challenging issues that do not occur in simulations. More specifically, building real systems forces you to handle cases where theoretical models are incorrect, or do not capture the right details. Thus realistic evaluation of technologies and their mutual interactions play a major role in identifying the key performance bottlenecks.

On the other hand, real testbeds do not allow repeatability and tight control as well as scalability, mainly caused by high costs for hardware, software, and manpower. To overcome these drawbacks, some research-oriented real network testbeds that support the development, debugging, and the evaluation of new network services were deployed: Emulab,[17] APE,[18] ORBIT,[19] RON,[20] PlanetLab,[21] PlanetLab Europe,[22] OneLab,[23] and GENI.[24]

Emulab is an interesting example of a real network testbed with public facilities. It is an experimental integrated platform that provides access to a wide range of real experimental environments like live-Internet (interaction with RON and PlanetLab), mobile wireless networks (six robots are equipped with sensor capabilities), sensor networks (25 motes are available), 802.11 Wireless, etc. In addition to network testbeds, Emulab also allows the execution of emulated experiments, as well as simulations. In particular, by using NS-2's emulation facilities,[25] Emulab allows the interaction of simulated

[16]Netkit Network Emulator. `www.netkit.org/features.html/`.
[17]Total network testbed. `www.emulab.net/`.
[18]Ad Hoc Protocol Evaluation testbed. `apetestbed.sourceforge.net/`.
[19]Radio grid testbed. `www.orbit-lab.org/`.
[20]Resilient Overlay Networks. `nms.csail.mit.edu/ron/`.
[21]PlanetLab platform. `www.planet-lab.org/`.
[22]PlanetLab Europe platform. `www.planet-lab.eu/`.
[23]OneLab European platform. `www.one-lab.org/wiki/view/OneLab`.
[24]Global Environment Network Innovations, `www.geni.net/`.
[25]NS-2 Emulation Extensions. http://ns-2.blogspot.com/2007/05/ns-2-emulation-extensions.html.

networks with real networks. Thus Emulab unifies different environments under a common user interface, integrating them into a common platform. NS-2's emulation facilities include methods and extensions allowing the introducion of the NS-2 simulator into a live network and simulating a desired network between real applications in real time. In particular, special objects within the simulator are capable of introducing live traffic into the simulator and injecting traffic from the simulator into the live network.

The original Emulab was primarily installed at the University of Utah, which is also home to most Emulab software development. Emulab, however, is now present in more than two dozen sites around the world.[26] Some Emulab testbed are the Deterlab (cyber-DEfense Technology Experimental Research laboratory) testbed at U.S.C. and at U.C. Berkeley[27]; the TIDIA/Kyatera Emulab testbed in Brazil[28]; the TWISC (Taiwan Information Security Center) emulation testbed[29] in Taiwan, etc.

Started as a research project at Uppsala University and partly funded by Ericsson, APE is the only existing testbed for large mobile ad hoc networks testbed (see footnote 18). APE contains an encapsulated execution environment or, more specifically, a small Linux distribution and tools for post testrun data analysis. It aims at making the process of performing complex real-world tests as easy as possible. It focuses on smooth deployment, high ability of customization, and ability to easily run several routing protocol implementations for comparisons.

ORBIT (see footnote 19) is a radio grid testbed for the evaluation of next-generation wireless network protocols. ORBIT consists of a 400-node indoor radio layer emulator, with 64 static nodes in a grid layout equipped with wireless network interfaces, and a 50-node outdoor, full-scale network. Funded by the the National Science Foundation, ORBIT is a collaborative effort between several university research groups in the New York and New Jersey region and industrial partners like Lucent Bell Labs, IBM Research, and Thomson.

The MIT Resilient Overlay Network (RON) (see footnote 20) is a large platform funded by DARPA. Consisting in an application-layer overlay on top of the existing Internet routing substrate and having 17 sites located around the Internet, RON allows research in Internet-based distributed systems (i.e., resilient routing, peer-to-peer systems, distributed application development, etc.). Related testbeds are the X-Bone project,[30] which provides a toolkit for rapid deployment of overlay network for things like IPv6; and the 6bone testbed,[31] an IPv6 testbed to assist in the evolution and deployment of the next-generation Internet network layer IP protocol.

The development of the PlanetLab platform (see footnote 21) had as a key motivation to evolve an improved Internet architecture by implementing new protocols as an Internet overlay. More specifically, PlanetLab is a network of computers located at sites around the world, forming a testbed for creating and deploying planetary-scale services. In this way, PlanetLab serves as a testbed for overlay networks and responds to the interest of the research community in experimenting with large-scale applications. New large-scale services can then be tested and validated in an environment that is intended to replicate the environment of the Internet but does not disrupt the performance of the Internet.

A PlanetLab independent slice and management authorities spanning Europe is emerging: PlanetLab Europe (see footnote 22). Its control center is in Paris and it is also

[26]Other testbeds. www.emulab.net/docwrapper.php3?docname= otheremulabs.html.
[27]Network security testbed, www.deterlab.net/.
[28]www.emulab.lasc.usp.br/index.
[29]testbed.ncku.edu.tw/.
[30]www.isi.edu/x-bone/.
[31]go6.net/ipv6-6bone/.

federated with the worldwide PlanetLab control center in Princeton. This guarantees the access of the entire worldwild PlanetLab platform to joint members of PlanetLab Europe.

PlanetLab Europe is in fact supported by the OneLab (see footnote 23) project, financed by the European Commission. OneLab is an open networking laboratory that has as its goal to extend the PlanetLab infrastructure by enabling the deployment of PlanetLab nodes in new wireless environments. In addition, OneLab also aims at the improvement of monitoring capabilities of PlanetLab, taking into account both networking and system performance issues.

Finally, the Global Environment Network Innovations (GENI) project (see footnote 24) consists in having a bold new research platform, supported by the American National Science Foundation (NSF). GENI aims to help the construction of a 21st century Internet, a new Internet fundamentally better than the Internet of today. The goal of GENI is to enhance experimental research in networking and distributed systems, and to accelerate the transition of this research into products and services that will improve economic competitiveness. The GENI platform will consist of a collection of physical networking components, including a dynamic optical plane, forwarders, storage, processor clusters, and wireless regions.

Emulab, APE, and ORBIT are some examples of testbeds allowing experiments in WSONs, while RON, PlanetLab, PlanetLab Europe, OneLab, and GENI focus on Internet-based experiments. In particular, Kropff et al. [2006] and Kiess and Mauvea [2007] have provided interesting surveys on real-world experiments and testbeds of ad hoc networks. These documents have reported on key attributes of some testbeds and provided a testbed classification to aid researchers in selecting the appropriate candidate tools for their experiments.

4.4. Prototyping

Prototyping is the process of building a model of a system. This model is then used to test the designed aspects or to illustrate ideas of the system. In this way, a prototype represents a useful way to refine designed ideas as a preparation for the final system deployment.

Recently, the new concept of *rapid prototyping* has emerged as a powerful tool to evaluate protocol performance. Abdesslem et al. [2007] introduced the Prawn (PRototyping Architecture for Wireless Networks) software environment. Prawn allows rapid prototyping and focuses on obtaining, with little effort, an instantiation of the system that may not be optimized but is fully functional and complete. The goal is thus to make prototyping become attractive for designers by making systems design and evaluation as easy, quick, and effortless as possible. Notice that prototyping represents a different but complementary process of protocol design compared with real testbeds.

Prawn provides a set of basic building blocks (like neighbor discovery, link quality assessment, message transmission, etc.) that implement common functionalities required by wireless protocols. By using a language-independent API, the designer can then use the defined high-level primitives to send and receive data, and retrieve information from the network, without caring about sockets, communication setup, addresses, etc.

Contrary to real testbeds, rapid prototyping addresses very early stages of the design process. It server as a complementary approach to simulation, emulation, or real testbeds. In opposition to the previous nonformal approaches, the main goal of rapid prototyping is to facilitate the protocol design process.

5. FORMAL APPROACHES

By surveying the literature, it appears that protocols most of the time are designed using rather nonformal approaches. In the initial phase, nonformal approaches basically consist of a textual description of services and data structure provided by the protocol. This design process is then iteratively carried on through incremental phases. During these phases, the designers informally refine the protocol by appending details and checking eventual errors. Finally, once the designers are satisfied by the service requirements and convinced that the protocal is no longer errorprone, the design process is terminated and the protocol is provided. Nevertheless, as this design process is based on informal refinements, it is impossible to guarantee that all requirements are globally respected. In this manner, and for many protocol standards (e.g., those of ITU, IETF, etc.), these designed specifications contain ambiguities and omissions. Even if a programmer implementing the service attempts to reasonably resolve the ambiguities, it will in all likelihood lead to an incorrect and inconsistent implementation. That is the reason why many efforts conducted by the formal description techniques (FDT) community as well the mathematical analysts community are still devoted to formally modeling and validating the protocols.

Due to the flexible nature of wireless services and the large size of their user community, errors not detected through the incremental design process described above, and this could have a great impact, especially if problems appear once the systems are deployed. In addition, although analytical methods for modeling and analyzing complex systems have been known since the 1960s [Petri 1962; Markov 1971] and though formal approaches have been used to validate wired systems for many years now [Holzmann 1991], wireless self-organizing networks raise many novel constraints that open new issues in the design, analysis, and formal validation of their protocols (see Section 3). Nevertheless, some formal methods and tools may be successfully adapted to be used in the wireless context. Contrary to nonformal approaches, the main objective of formal approaches is to provide a nonambiguous model of the protocol from which quantitative or qualitative properties may be extracted and studied.

Formal models can be defined using high-level formalisms (for example SDL, PROMELA, stochastic Petri Nets, etc.) or more low-level formalisms that are often the underlying semantic models of the high-level ones (for example, finite automata, labeled transition systems, Markov chains, etc.). According to the type of properties to be checked, the appropriate formalism and its associated underlying model have to be properly chosen. Indeed, they share drawbacks that are stated in the Section 6. While, some formalisms are more convenient to use in addressing the broadcasting feature of WSON such as process algebra, others are more efficient in dealing with the table-driven features such as SDL or PROMELA. We do not detail in this article the languages that are more adapted to each feature; instead the major target here is to discuss validation techniques relying on formal approaches.

The main goal is then (1) to evaluate the performance and the dependability of a system (by the mathematical analysis community), (2) to match the formally specified and extracted properties with the protocol requirements or directly with the implementation, by reasoning about the system using mathematical relations. First, the mathematical analysis community provides diverse techniques to model complex systems applying mathematical aspects. The main goal here is to evaluate the performance of a system. Second, two main technique sets define the formal validation approaches that are commonly applied to network protocols: *verification* and *testing*. The goal of formal verification is to improve reliability by arguing through mathematical logics and thereby to check that formal system model fully complies with a given set of requirements. Regarding formal testing, test sequences are generated from the correct and verified models in order to be checked on real implementations. Formal approaches are thus able to guarantee that a protocol is efficient and free of errors and that the implementation satisfies the requirements.

These formal communities provide nonambiguous and correct formal models as well as implementations conforming to informal requirements. They cover the broad cycle of protocol development and drastically enhance its reliability and efficiency. These techniques are, however, quite distinct and hence constitute three different communities.

5.1. Formal Mathematical Analysis

Models are used to answer questions related to efficiency and trustworthiness, offering a means to comprehend an otherwise incomprehensible problem. It helps to visualize the problem, to break it down into discrete, manageable units. Although it is a simplified abstract view of the studied complex system being studied, a model has the features required to accurately capture the behavior of the system.

The performance of a system can be improved via modeling it using mathematical methods and by reasoning about it. Mathematical models can be classified as: linear versus nonlinear, deterministic versus probabilistic (stochastic), static versus dynamic, or lumped versus distributed parameters.

Among the mathematical models, stochastic models have been found useful in the design and analysis of advanced computer and communication systems. *Stochastic* is synonymous with *random*, which means that, in a stochastic model, randomness is present, and variable states are not described by unique values, but rather by probability distributions. A stochastic model can thus be used for estimating probability distributions of potential outcomes by allowing for random variation[32] in one or more inputs over time. This means that, even if the initial inputs are known, there are many possibilities the outcomes might go to, but some paths are more probable and others less. Distributions of potential outcomes are derived from a large number of simulations (stochastic projections) which reflect the random variation in the input(s). This large number of simulations are usually gotten by stochastic techniques called Monte Carlo methods [Doucet et al. 2001; Rubinstein and Kroese 2007; Kwon and Kim 2008; Wang and Xiong 2005]. A Monte Carlo method is a technique for iteratively evaluating a deterministic model using random numbers and probability to solve problems. It is just one of many methods for analyzing uncertainty propagation, where the goal is to determine how random variation, lack of knowledge, or error affects the sensitivity, performance, or reliability of the system that is being modeled.

There exist several high-level stochastic models, for example, stochastic process algebra, such as the Performance Evaluation Process Algebra (PEPA) of Jane Hillston,[33] the Stochastic Automata Network (SAN) [Plateau and Atif 1991], stochastic Petri nets, etc. The Petri nets, in particular, developed in the early 1960s by C. A. Petri in his Ph.D. dissertation [Petri 1962], are useful for modeling concurrent, distributed, asynchronous behavior in a system [Peterson 1977, 1981]. Also known as a *place/transition net* or *P/T net*, a Petri net is a directed bipartite graph, in which the nodes represent transitions (i.e., discrete events that may occur), places (i.e., conditions), and directed arcs. Arcs run between places and transitions, never between places or between transitions. The places from which an arc runs to a transition are called the *input places of the transition*; the places to which arcs run from a transition are called the *output places of the transition*. For nonfunctional properties, timing concepts need to be added. To achieve this, a firing delay is associated with each transition that represents the time at which the transition has to be enabled. In the case of a delay associated with a random distribution function, we call the resulting net class a *stochastic Petri net.*

[32]The random variation is usually based on fluctuations observed in historical data for a selected period using standard time-series techniques.
[33]`http://www.dcs.ed.ac.uk/pepa`.

The common underlying semantic model of such stochastic high-level models is the Markov chain. Markov chain models are some of the most powerful tools available for analyzing complex systems and are derived from the Markov property [Markov 1971]. This property states that "given the current state of the system, the future evolution of the system is independent of its history." In other words, a Markov chain is a sequence of random variables $x_1, x_2, x_3, \ldots$ with the Markov property, namely that, given the present state, the future and past states are independent.

Markov chains are often described by a directed graph, where the edges are labeled by the probabilities of going from one state of a system to the other states. In discrete-time Markov chains, the changes of a state of a system are only performed in discrete time. On the other hand, in continuous-time Markov chains, changes can happen in any random exponential distributed time. A property allowing the description of a continuous-time Markov process in an equivalent discrete version can be also found in the literature, allowing the simple and efficient definition of interest measures [de Souza e Silva and Gail 2000; Bolch et al. 2006].

In queueing theory, Markov chains are also used to model the system in each state [Bolch et al. 2006; Kleinrock 1975]. Queueing models allow a number of useful steady-state performance measures to be determined, such as the average number in the queue or in the system, the statistical distribution of those numbers or times, the probability the queue is full or empty, and the probability of finding the system in a particular state [Leao et al. 2001; Elwalid and Mitra 1992]. Another high-level formalism, the stochastic fluid model, has also been largely used in the literature for analyzing different computer and communication systems [da Silva et al. 2004; Kleinrock 1975]. In particular, when the events rate of a system vary by many orders of magnitude, the use of fluid models can result in a considerable reduction of the computational cost when compared with the models where all the events are explicitly represented.

The main problems in modeling networks are that models are usually too large to be handled by a computer system, and, due to model complexity, model development is very time consuming. As a solution, methods have been developed for complexity reduction, thus considerably reducing the development time [da Silva 2006].

Regarding the domain of WSONs, some researchers have used analytical models for modeling medium-access protocols in wireless networks. The models are thus used to compute throughput of flows in arbitrary network topologies or specific topologies [Bianchi 2000; Chaudet et al. 2004; Ray et al. 2005; Boorstyn et al. 1987; Garetto et al. 2006, 2008b], to analyze delay in single-hop 802.11 networks [Tickoo and Sikdar 2004] and to study the capacity in mobile ad hoc networks and DTNs [Garetto et al. 2007, 2008a, 2008b].

Finally, all the works discussed in this section dealt with quantitative properties in a formal manner in order to evaluate performances. Once the formal model and formal reasoning are performed, the simulation techniques presented in Section 4 may be used as a complement to analyze the quantitative properties.

5.2. Verification Techniques

There exists very little research on the validation of wireless systems and more specifically on WSONs. The reason is twofold. First, many researches believe that the inherent constraints of such systems make the verification process very time consuming and that an important threshold has to be crossed before doing so becomes proficient. Second, it was shown that there are inherent limitations on the methods for such wireless protocols [Wibling 2005]. Nevertheless, we cite some of the studies providing techniques to formally verify protocols in wireless self-organizing networks.

In order to formally validate a protocol, the protocol has to be initially specified in an unambiguous and structured manner. Two main approaches are advocated for the verification.

—The first, called *code verification*, or *static analysis*, means that the protocol is implemented with a regular programming language such as Java and the code is verified afterwards. It becomes common that model checkers work directly on the source code of software implementations rather than on a model provided by a user or a standard. We can the model checkers BLAST and SLAM [Eisner 2005; Gunter and Peled 2005], which work on C source code. For Java programs, we mention Bandera [Corbett et al. 2000] and Java PathFinder [Havelund and Pressburger 2000], and for C++ programs Verisoft [Godefroid 1997]. In such techniques which consist in constructing a model from the source code, the abstracted model could lose some information and the code still contain ambiguities, and flaw. As far as we know, even if static analysis is widely used in other areas, it has been used only once in regard to WSON protocols [Engler and Musuvathi 2004]. The authors directly analyzed the code for finding errors in different AODV implementations [Perkins et al. 2003]. This method has been utilized especially for bug assessment though it allows identifying a loop error in some of the implementations.
—The second approach consists of describing the protocol using formal description languages. These may be subsets of logics (first order predicate logic, for instance) or subsets of automata (or transition systems) representations.

Commonly, the requirements of the system that have to be verified are expressed using temporal logics. Qualitative properties may be designed by CTL [computation tree logic; Clarke and Emerson 1981], LTL [linear temporal logic; Pnueli 1981] or TCTL [timed computation tree logic; Alur and Dill 1990]. These properties may be categorized as either *liveness* or *safety* [Kindler 1994]. A liveness property means a true property that will eventually happen in the future, and a safety property specifies a "bad" property that never happens. Although safety properties can be noticeable in a finite running time of the system, liveness properties require sometimes infinite system runs. In the same way, quantitative properties are often specified using specific quantitative models such as PCTL [probabilistic computation tree logic; Emerson 1990; Hansson and Jonsson 1994]. But we may also cite probabilistic timed automata [PTA; Kwiatkowska et al. 2002a] (based on Markov chains or processes), quantitative transition systems [QTS; de Alfaro et al. 2004], quantitative bound μ-calculus or quantitative-bound automata (a combination of a quantitative automaton and bound functions) [Chakrabarti et al. 2005].

From those formal models and the expressed properties of the requirements, two main approaches are applied. The first is an algorithmic verification method commonly called *model checking* [Clarke et al. 1999], which has had a broad success in the industry, especially due its interesting results (the possibility off easily finding flaws and helping correct them) as well its wide toolset. The second method, called *deductive verification* [Manna et al. 1999], consists of formally proving that a property may be drawn from a given set of premises. While the main advantage of this latter approach is to prove properties of infinite state systems for which theorem provers are processed, the advantages of model checking are to obtain a verdict in an automatic way and quickly locate the flaws.

Figure 4 shows a summary of the verification techniques and their order of use.

5.2.1. Model Checking Approaches. Model checking approaches have been more commonly used in wireless areas than deductive verification approaches. Besides, both

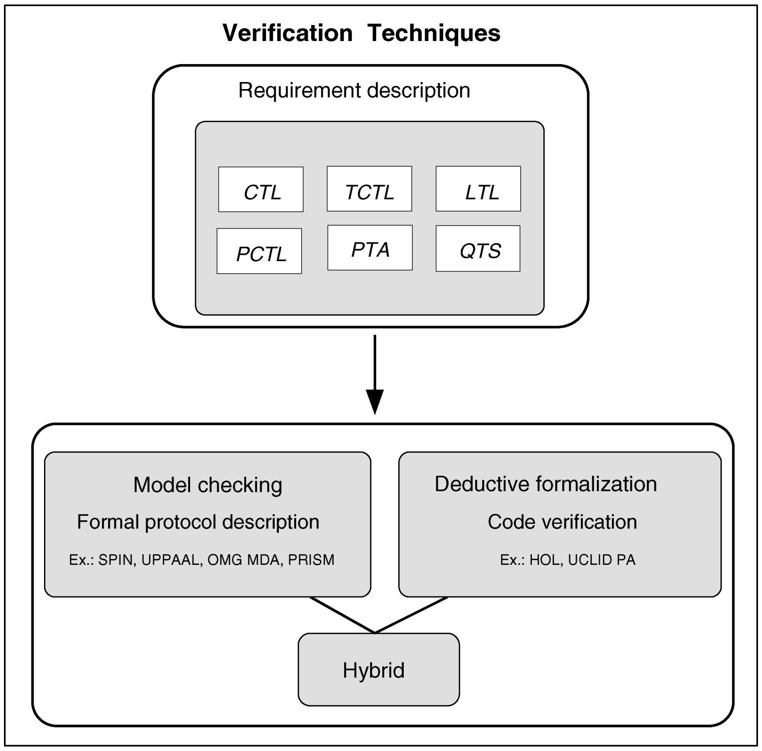

Fig. 4. Verification techniques.

qualitative and quantitative properties have been studied through diverse model checking approaches. We depict in the following some interesting applications to WSON protocols concerning first qualitative aspects and then quantitative ones.

de Renesse and Aghvami [2004] have utilized the model checker SPIN [Holzmann 2003] in regard to the MANET routing protocol WARP. By specifying their protocol using the high level-language PROMELA (PROcess MEta LAnguage) and their properties in LTL, the authors have covered 98% of the state space. However, the experiments were performed for a rather small network containing only five nodes.

A Bluetooth location system was also verified using SPIN and PROMELA in which the network topology consisted of 18 wireless nodes with various scatternet layouts and properties considering four basic properties [Fernández-Iglesias et al. 2005]. The authors illustrated an ill-defined protocol as well as incompatible properties on some of the configurations. By that way, even if very few properties were studied, it points to the possibility that topologies, mobility, interference, etc., could be real issues when defining such properties in WSON.

A formal approach to verify a model health service platform has also been performed by Jones et al. [2004]. The main contribution of this research was to integrate the OMG MDA (model-driven architecture) concepts into the validation process. Although the results were few and the study had a limited testing coverage, it raised challenges in verification when using models processing by means of MDA.

The model checker UPPAAL [Larsen et al. 1997] was applied by Chiyangwa and Kwiatkowska [2005] in order to study the timing properties expressed in LTL of the MANET routing protocol AODV. Interesting results and possible enhancements of the protocol were provided. However, the experimented topology was linear and a restricted number of nodes were chosen.

The UPPAAL tool plus TCTL expressions were applied to verify the quality of service on a MAC protocol for wireless sensor networks [Watteyne et al. 2005]. The monitored topology considered these was linear and the main properties verified on the protocol were real-time constraints especially for automotive domains. Despite some important given assumptions (particularly on the topology), an exhaustive exploration of all paths of the system execution graph was conducted in which interesting scenarios were executed. The results especially on the timing constraints verification in such networks were very promising even if issues still remain. Complementary results for their verification process were obtained by Godary et al. [2004], where UPPAAL was evaluated.

Regarding quantitative verification or (bound) model checking in WSON, little work has been performed up to now. Mainly, the quantitative properties that are verified by model checking have been the system power requirements, system lifetime, response time, peak running total, flooding, dynamic power management, dynamic voltage scaling, and real-time delays. We cite the following studies.

In Kwiatkowska et al. [2002b], model-checking was used for the verification of the 802.11 standard. The authors focused on assessing the performance of the basic access (a distributed coordination function) and two-way handshake achieved through the verification of properties expressed in PCTL temporal logic. For that purpose, the authors constructed a probabilistic model (a Markov decision process) referring to a specific and fixed topology consisting of two source and two destination nodes. Nevertheless, due to the complexity of the study and the size of the probabilistic timed automata, the scalability is uncertain.

Ballarini and Miller [2006] described the use of probabilistic model checking for the comparative analysis of an IEEE 802.11 set of protocols for MANETs with S-MAC, a protocol designed to reduce energy consumption suitable for wireless sensor networks on a 3-hops topology. The formal specification is a Markovian model which is analyzed using a probabilistic model checker like PRISM [Rutten et al. 2004]. The model verification was achieved through a combination of specific probabilistic reachability PCTL properties and rewards properties. However, the main drawback (common to much other research) is still the scalability, the topology applied in that article being indeed relatively limited (four nodes, one source, one sink).

Fehnker and Gao [2006] presented a performance analysis of flooding protocols in wireless sensor networks applying probabilistic verification. The authors used both the PRISM model checker with Markov chain processes and the Monte-Carlo simulation. The checker allowed them to analyze performances of the protocols even applied on nondeterministic models.

Finally, in Kwiatkowska et al. [2005], the PRISM model checker was also used, but applied to dynamic power management properties. Probabilistic labeled transitions and PCTL are designed to study the performance of dynamic power management policies and dynamic voltage scaling schedulers.

5.2.2. Deductive Verification Approaches. Deductive verification techniques are less frequently applied. We have found very little analysis regarding qualitative properties and, to the best of our knowledge, there exists no related work on quantitative properties in WSON. The main reason for this may be that the theorem prover tools are not self-sufficient and, because of the complexity of the constructs or functions designed, most of the time experts are required to manipulate such tools. Nevertheless, we cite a few studies regarding WSON in this area.

Bhargavan et al. [2002b] applied the HOL [HOL 2005] theorem prover in their deduction of route validity and freedom from routing loops in AODV. They utilized three conditions on next-node pointers, hop counters, and sequence numbers to form a path (namely, the *invariant* in deductive verification) on pairs of nodes (on the path from source to destination). Three properties were verified using SPIN, after which HOL allowed than to prove that the three properties led to the path-invariant theorem. But an important contribution was to illustrate that it's quite difficult to specify WSON protocols. Indeed, despite the interesting results provided by the authors, they noticed themselves that they failed to prove properties for AODV (while they did prove them for RIP).

Another work in regard to the MANETs proved the absence of routing loops in a simplified version of AODV [Das and Dill 2002]. The strategy is quite similar to the previous one, but more automated. They used predicate abstraction and could raise most of the quantified predicates automatically by analyzing spurious counterexample traces. The method successfully discovered all required predicates for the version of AODV considered but in a rather limited network that had with three nodes and was without any mobility.

Following the previous example, a tool called UCLID PA [Lahiri and Bryant 2005] was used to prove the freedom of loops in AODV once again [Lahiri 2004; Lahiri and Bryant 2007]. While the previous verification process was done using quantified predicates, the these authors used indexed predicate abstraction, therefore providing an enhanced axiomatic system but applied on the same very small and limited network.

Borujerdi and Mirzababaei [2004] verified properties extracted from the MobiCast protocol [Tan and Pink 2000] integrated into a mobile network with microcells. The protocol has been formally specified in Prolog language [Sterling and Shapiro 1994] and the expressed properties proved by applying an SLD resolution [Spivey and Spivey 1996]. Several inconsistencies were detected in the first version of the protocol and resolved with minor variations. And despite the fact that no real mobility was applied to their experiments, the authors demonstrated that formal approaches are required all along the development phase of a protocol.

5.2.3. Hybrid Approaches. Although model checking and deductive methods are commonly employed separately, an attempt to merging both techniques in wireless network domain has been studied. McIver and Fehnker [2006] defined a stepwise specification and refinement of WSON protocol characteristics using a combination of proof-based methods and model checking. Energy-efficient protocols in wireless networks have been considered especially by analyzing delays and collisions criteria in such systems. From an exhaustive search with model checking, the authors illustrated weaknesses in the systems and thus provided lower and upper bounds on quantitative aspects of the protocol. Formal proofs also enabled them to investigate optimal protocol behavior especially in terms of trading off energy requirements and performance. Nevertheless, both methods being quite different, problems regarding the obtaining of realistic formal models and performance criteria for such WSONs were raised.

5.3. Formal Testing Techniques

Formal techniques for protocol testing have been performed for a long time now [Holzmann 1991]. Two main mechanisms may be utilized for this purpose: *passive* and *active* testing.

Active testing is based on the execution of specific test sequences in regard to the implementation under test. These test sequences are obtained from the formal model according to coverage criteria. These criteria can then be applied on the specification, for example, coverage of all logical conditions, coverage of all paths. This allows setting if the specification as well as the code was covered during the testing phase.

The tests may be generated automatically or semiautomatically from test criteria, hypothesis, and test goals. The format of these sequences commonly used by the testing community is TTCN3 [ETSI 2007], from which their executions are performed through points of control (execution interfaces). These points of control are installed in the context of a testing architecture, which means the way to put the testers (e.g., upper and lower testers to test a specific stack layer, the different interfaces, and the oracle in order to provide a verdict on the executed tests to the tested implementation). We can mention two families of testing: the *static* and *dynamic for active testing*. The first is based on static analysis of the source code, that is, the implementation. The code is inspected regarding the checklist elaborated or by analyzing the control and data flow graph. Using this kind of test, we do not have to exercise the system under test with real data. On contrary, dynamic testing implies that the system under test is executed under different configurations, that is, with different input data tests. The test sequences to be exercised on the implementation are derived from the model described by an FDT. Afterwards, the inputs of the test sequence are given to the implementation and the output results are compared to those expected by the specification.

Passive testing consists in observing the input and output events of an implementation under test in runtime. The term *passive* means that the tests do not disturb the natural runtime of a protocol or service. This concept is sometimes also refereed to as *monitoring* in the literature. The record of the event observation is called an *event trace*. This event trace will be compared to the formal specification as a test sequence. The passive testing techniques are applied especially because the active ones require important testing architectures, which the testers need to control the system at some specific points. This is sometimes not feasible or even undesirable. Nevertheless, while test sequences in active testing may give concrete verdicts (except for “inconclusive” ones), an event trace that satisfies the model does not mean that the whole implementation satisfies the specification. On the other hand, if a trace does not satisfy the specification, then neither does the implementation.

Passive and active testing have their own advantages and drawbacks, especially when used on wireless self-organizing network protocols. Nevertheless, the results obtained depend on the system under test and essentially on the testing goal, the testing type. The testing type considers the whole testing process of the protocol, which consists of different steps: unit, conformance, interoperability, integration testing, and so on. Most of these testing types are normalized. For instance, conformance testing has been standardized by the ITU-T in Technical Report 9646 [ISO 1994], in which common testing architectures, interfaces, or points of control and observation are mentioned and specified. Nevertheless, these standards are mainly designed for wired systems and most of the time (if not always) the new inherent constraints of WSONs (such as the lack of infrastructure or noncentralized systems) are omitted from these documents. Although the tasks for testing in such an environment seem tough, studies on formal techniques for testing protocols in such wireless contexts are presented in the following paragraphs.

Figure 5 shows a summary of the formal testing techniques.

Lin et al. [2003] proposed a formal methodology to specify and analyze a MANET routing protocol. It was based on the relay node set (RNS) concept. An RNS is a set of nodes allowing all nodes in the network to be reached. According to the protocol, the set is built differently: the reactive protocols build the set in a regular manner, when it is required, while the proactive ones build the set during the route discovery performed at the beginning of the network lifetime, being maintained during the network lifetime. Passive testing techniques for conformance testing are applied in that article.

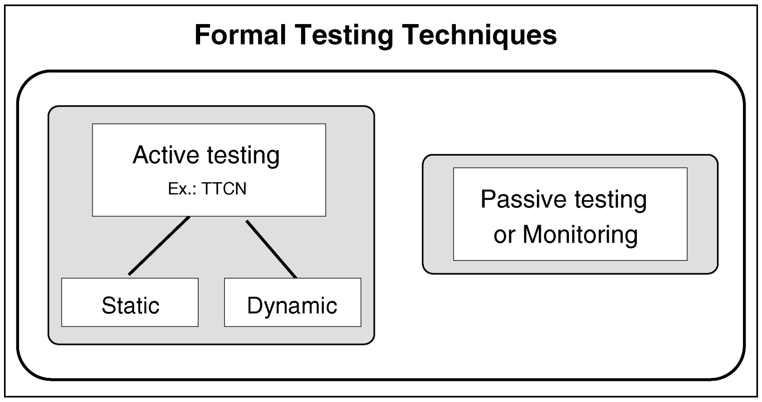

Fig. 5. Formal testing techniques.

The framework illustrated in it has as its main goal to analyze the implementation under test by metrics through nonfunctional aspects which are usually applied for a performance analysis.

Glässer and Gu [2005] developed a formal model, namely, Distributed Abstract State Machines (DASM), to allow the specification of ad hoc network routing protocols. Their motivation was also to raise the main issues when designing such protocols. Nevertheless, only conformance testing by an active mechanism has been processed. While the model allows verifying the behavior of a node in a functional way, it is nonexecutable and does not allow observing of the interoperability of nodes.

Zakkuidin et al. [2005] provided another approach to formalizing testing of a MANET routing protocol by applying game theory concepts. Game theory is based on the "income" calculus. *Income* means, for instance, the convergence when the topology is modified or the induced overhead. A strong hypothesis commonly applied in game theories consists of the complete knowledge by the "players" of the "game." It means that each node is supposed to have a complete knowledge of the network topology as well as of the nodes or link states. Furthermore, complete knowledge forbids the nondeterminism. Therefore, some mobility models based especially on nondeterministic behaviors and commonly used in WSON analysis can not be modeled. In many wireless self-organizing networks and because of their characteristics, the complete knowledge assumption may not hold, which invalidates the test sequence execution.

Maag et al. [2008] presented a new methodology for the conformance testing of a MANET routing protocol, namely, Dynamic Source Routing (DSR). Active mechanisms were applied combined with a nodes self-similarity approach allowing avoiding the so-called "state space explosion" issue and reducing the needed formal model to a dynamic topology. The approach was applied to the DSR-UU implementation by E. Nordström at Uppsala University.[34] Interesting ideas for interoperability testing contexts were discussed.

Merouane et al. [2007] proposed an interoperability testing approach based on an EFSM- (extended finite-state machine) based language. A testing architecture was provided and test sequences automatically generated from an SDL specification in order to test an implementation of DSR-UU. This work reveals the real complexity of studying the interoperability aspects of WSONs. Some of the systems were so dynamic that the observation or control of some real links (through the interfaces of nodes) became a hazardous task. The authors therefore employed an emulation mechanism

[34]DSR-UU v0.1. `http://core.it.uu.se/core/index.php/DSR-UU`.

coupling an NS-2 simulator and the implementation under test integrated into two real nodes.

5.4. Hybrid Techniques

Although these verification and testing techniques are quite different regarding their approaches and their own objectives, they are also complementary. That is the reason why interesting studies have appeared that present the advantages of mixing verification and testing approaches. Therefore, Maag and Zaidi [2007] defined a stepwise approach that uses two languages to specify the OLSR protocol [Clausen and Jacquet 2003]. They applied Promela associated with the powerful tool SPIN to perform verification and SDL associated with the ObjectGeode tool [Verilog 1997] to generate test sequences. Since both languages are different, the design approaches have not been the same and a compromise has also been required in regard to the specification choices. Finally, the work illustrates essential issues while covering the protocol lifecycle. Indeed, the authors revealed that the languages do not have the same power of expressiveness and that some aspects which can be shown in one form with one language could be impossible with the other. This is therefore why these validation activities are most of the time completely distinct.

6. OBJECTIVES, LIMITATIONS, AND DRAWBACKS

In the previous sections, we have presented nonformal and formal approaches commonly used by the network and the performance analysis communities to address nonfunctional properties. In the section devoted to formal approaches, some relevant work coming from the verification and formal methods communities were presented, which have been carried on to formally verify functional properties regarding the behaviors of protocols.

This section recalls the main objectives of the techniques used in formal and nonformal approaches. The limitations and drawbacks of such approaches are then enumerated and discussed with regard to the WSON domain.

6.1. Nonformal Approaches

6.1.1. Main Objectives and Techniques. Simulation, as explained previously, is a technique widely used to evaluate protocol performance and seems to give good results for a large number of protocols [Kiess and Mauvea 2007]. It models influencing factors and algorithms and investigates these aspects in an artificial software environment with a high degree of abstraction. This technique appears at the same level as the formal verification technique of the protocol development cycle. Simulation is also widely used because the manipulated concepts are easier to use to model a system than analytic analysis, which are based on mathematical equations. The main advantages of this technique are that it permits one to observe and analyze protocol performance and to verify expected properties (e.g., network lifetime extension, data delivery). Moreover, it is much cheaper than other techniques as it allows reproducibility and repeatability with a low effort just by changing some model parameters.

Emulation provides the best compromise between cost and precision. Emulation is a mix of software and hardware combined to reflect the behavior of a network. Some of the components are real and others are simulated. The main objective is thus to test protocols in a more realistic environment, as a way to better forecast and analyze protocol behavior, for later real-world implementation. This technique inherits the drawbacks of simulation that will be detailed below.

Real-world testing consists of performing tests in a real testbed, which is an implementation of the system very similar to the final one. The dimensioning and the configuration of real testbed experiments are, in general, hard and costly processes.

Since potential problems are only detected at the end of the lifecycle of the system, if a built system is not the expected one, the implementation phase should be restarted again from the beginning.

To summarize the techniques coming from the network research and performance engineering area, we can clearly establish that the common objective is to evaluate or to verify properties that are commonly named as nonfunctional properties, such as servers loads, response times, rates of loss, data delivery reliability, etc.

6.1.2. Limitations and Drawbacks

(1) Simulation techniques

—The main drawback that can be formulated against simulation techniques [Kiess and Mauvea 2007] is based on the assumptions that are made regarding the underlying protocol layers implemented in the simulators. More specifically, they represent potential errorprone points. In fact, models used by simulators have usually not been previously validated, which may add errors to the simulated environment. Hence this can impact the protocol performance evaluation and, finally, result in a false analysis.

—We can also mention that simulation, as explained in Wei et al. [2003], does not capture the internal behavior of a system. Indeed, some aspects of the system that are relevant to performance evaluation may not be directly observable, constituting then "black boxes." Hence performance evaluation related to unit measurement can be strangled by the "black box" effect of the system. To overcome the lack of observation, the designers can use a lower level of abstraction to model a system. But by reducing the level of abstraction, they can be too close to the real implementation.

—The main concern with simulations is, however, the level of confidence we can have in them. Some works in the literature show that divergences in obtained results exist between different simulators [Cavin et al. 2002] and that common existing it falls can impact the reliability of a simulation-based study [Kurkowski et al. 2005]. The main conclusions are as follows: "omitting detail or oversimplifying the simulation model can lead to ambiguous or erroneous outcomes" ([Andel and Yasinsac 2006], page 3; "simulation assumptions always affect research outcomes" ([Andel and Yasinsac 2006], page 5; and, finally, "insufficient statistical analysis of independent simulation runs and improper data collection techniques can produce ambiguous or inaccurate conclusions" ([Andel and Yasinsac 2006], page 5.

—Although nonformal approaches are highly suitable for achieving performance measures in large-scale networks, they cannot easily prevent protocol design errors. Indeed, there are properties of protocols that do not relate to performance.

—Packet loss is commonly observable and controllable by using simulation tools [Lundgren et al. 2002]. Nevertheless, the implementation of IEEE 802.11 in NS-2 is rather unrealistic because several properties (e.g., fluctuating links or different transmission rates) cannot be represented as they depend on the implementation of radio propagation.

—From the physical layer viewpoint, the reality (i.e., mobility and environment models) is really tough to describe. Furthermore, wireless propagation models in simulators are often too simple and general, being dependent on the details or granularity required and integrated into the simulators [Cavin et al. 2002]. Therefore, qualitative and quantitative divergences between the results obtained from different simulators might be observed even by applying the same scenarios and parameters.

(2) Emulation techniques
—The main concern with emulations is, however, complexity. Emulators are complex pieces of software, but valuable because of their ability to maintain a closer connection to the authenticity of the hardware device. Writing an emulator is also a difficult process which requires obtaining appropriate system information and then figuring out how to emulate the system hardware with software code. In addition, the computational resources needed to run an emulated device are typically higher than those available in the device itself. Consequently, the performance of an emulated device is, in general, lower than that of the real device. This may pose limits to the scalability of the size of the emulated network [Rimondini 2007].
—Emulators have to face timing constraints [Kiess and Mauvea 2007] that need to be thoroughly stated. Thus, due to these real-time constraints, the simulation parts can be rapidly overstepped, which can lead to scalability problems.
(3) Real-world testing techniques
—While real testing experiments are ideal to tackle errors invisible in simulation, experiments become rather limited as the number of nodes increases [Kiess and Mauvea 2007]. Furthermore, these experiments have to be carefully executed and orchestrated.

6.2. Formal Approaches

6.2.1. Main Objectives and Techniques. As previously described, there exist various techniques to design a protocol by means of formal methods. This section rapidly sketches the different techniques that are available and emphasizes their limitations, and how they step to in the development lifecycle.

In the area of formal methods, verification and testing techniques have been widely used for wired protocols, and are also used in WSONs. Section 5 identified several studies on these two main domains that tried to transpose well-known techniques to WSON protocols. Nevertheless, this is not a straight rearrangement due to the new challenges introduced by WSONs (see Section 3), which underline new limitations or exacerbate existing ones.

Verification consists first in providing a model of the system. At this stage, the model will be used to automatically verify some properties of the protocol on the model and not its performance. By this technique, we can verify the correctness of the protocol regarding the requirements of the desirable behaviors of the protocol. The model issued from this step can be used to generate the tests. The active testing generation phase has to deal with a model that has been first checked, that is, that has been determined to be conforming to its requirements. Furthermore, the model to generate the tests can be constructed only for testing purposes and cannot come from the verification phase. The languages used for test generation and for verification goals are often not the same. Nevertheless, the model from which the tests will be generated requires verification regarding it adequacy to meet requirements and also regarding some inherent properties in the protocol, such as being loop free, or deadlock free.

In comparison to verification by model checking, deductive verification can handle infinite-state systems with very rich data structures. The main drawback of this approach, which can explain that its use is not widespread, relies on the design by the user of constructs, such as invariants and ranking functions. In other words, it requires very accurate user skills to guide the theorem prover tool as the process is not fully automatic.

The techniques, which operate to generate test cases and to verify some properties expressed by logical formulas, are based on exploration of the state space of the model. Both techniques often have to face the well-known problem of state space explosion. Test generation methods produce test cases that are called *abstract tests*. To be executed on a real implementation, these tests have to be expressed in a concrete way, notably in the programming language used by the implementation, in order to be able to interact with it. During the execution step, the concrete tests are executed by feeding the implementation with the input of the test case and the answer of the implementation is compared to the expected answer. This way to communicate with the system is not always feasible and, to overcome this drawback, we mention passive testing techniques. Indeed, passive testing needs only to retrieve traces of the real implementation by means of *sniffers* and seeks on these traces the expected behaviors of the implementation.

While nonformal approaches are essentially dedicated to nonfunctional properties, the traditional concerns of formal techniques are to establish the correctness a system regarding its expected behavior. We identify these properties as functional, which, in regard to communicating systems, are related to the system behavior in terms of interactions.

6.2.2. Limitations and Drawbacks

—Modeling is a time-consuming activity that requires a tremendous effort, but that cannot be avoided at the model checking and model-based testing phase. Although constituting an essential phase to establish the correctness of the protocol, modeling is a very complex activity, especially regarding the level of abstraction to be considered.

—In order to address the new challenges raised by WSON, formal methods have to face the well-known problem of state space explosion. The explosion can occur when the state space storage grows exponentially with the size of the model. To overcome this problem, many techniques have been proposed such as symbolic representation, partial order reduction, compositional reasoning, abstraction, or symmetry [Holzmann 2003]. Nevertheless, we are still faced with this problem. This explosion is also responsible for the nonexhaustivity of the tests.

—Existing modeling languages have to deal with new features of WSONs. Whatever the techniques used, the language needs to be expressive enough to capture all the WSON features. Moreover, a tool is needed in order to automate verification and test generation of the model. It also has to be able to handle the logic used to express properties to be checked and also to cope with the language used to express test objectives. This aspect is very important in particular for active testing techniques.

—Languages used by verification techniques are not able to handle the broadcasting mode of wireless communication. Many languages used by verification do not allow performing such a type of communication. The way to perform such broadcasting could be to specify it by a unicast link with a matrix that contains the identifier of all the nodes around the network. This is still an open issue [Wibling 2005].

—Many models consider fading links with distance between receiving or sending nodes. Nevertheless, message losses due to collisions in a communication between in-range nodes are most of the time obscured by the model. In addition, the collision model is commonly required to challenge particular fault-tolerance mechanisms [Watteyne et al. 2005].

—While models for formal verification may easily describe timing aspects designed for WSON protocols, languages dedicated to testing approaches are not always (or even most of the time) adapted. Constraints such as the execution on a real system might suppose that the nodes are synchronized, which is a WSON challenge [Lundgren et al. 2002].

—The mobility of nodes in WSONs generates topology changes. For modeling and verification, we have to consider such changes of connectivity. This features related to the kinetics of nodes can increase the state space explosion and there is no straight way to address it. Nevertheless, Maag and Zaidi [2006] explained a way to cope with this drawback.

—Concerning the passive testing approach, we have to face a lot of inconclusive verdicts when the collected traces are too short. In this case, the reasoning about correct or faulty implementations is not possible. Moreover, the way to characterize the properties that are found on the traces are known as *functional patterns*. They characterize the expected behavior of the system. They have to be expressed in an efficient manner, that is, in a nonambiguous way. This activity relies on a strong expertise in regard to the system to be tested. Nevertheless, to overcome this difficulty some studies have tried to assist this step by providing automatic mechanisms to verify the soundness of the patterns on a formal model [Bayse et al. 2005]. Another drawback of such techniques relies on the difficulty of merging the offline traces of a multihop wireless network. A recent study [Zaïdi et al. 2009] has been done on the correlation of traces between two distant entities. For the WSONs, so far as we know, there is no research that shows how to correlate traces coming from different nodes in order to verify global properties.

7. DOMAINS CONVERGENCE

In practice, little cross-fertilization exists between the formal and nonformal domains, which often work separately. Nevertheless, several attempts in the research area of wired protocols have been made to converge these two domains, usually accomplished by researchers coming from the formal area [Bütow et al. 1996]. This article thus emphasizes the research conducted to make them converge. This section reports on this research and discusses the common required effort to achieve that cross-fertilization for the validation of WSON protocols.

The observations that have led to such research have been based on practical experiences. Indeed, design experts communicate rarely with performance experts. The obstacle can be explained not only by the difference in the concepts that they manipulate. Formal design methods and performance design do not address the same objectives. The first is devoted to establishing the correctness of problems by revealing errors. Otherwise, performance evaluation allows first avoiding system malfunctions due to overcongestion of resources [Cavalli et al. 2004].

Performance evaluation for software systems allows efficiently characterizing the right configuration of a system to respect a well-defined quality of service. Moreover, we can identify congestion or faulty management resources. As mentioned in Wei et al. [2003], designers have often neglected performance engineering. Discovering a performance problem very late in the development lifecycle of a protocol can be costly. Cavalli et al. [2004] stated that 80% of client/server systems have to be rebuilt because the low performance obtained does not meet requirements. In this way, software performance models of at an early stage can decrease performance failures. Indeed, the performance of a system can be strongly related to the architecture of the system. Ideally, the performance should be considered at a early stage of the design of the system.

After these statements, in the following sections, we address the attempts of convergence in several manners and at different levels. We first report on the works performed at the modeling step, that is, on the description of what the system is expected to do. In the formal area, the model is called a *formal specification*. The way to take into account performance requirements at the specification level consists of annotating the model with performances requirements. Afterwards, we report on testing experiences.

We explain how, for correctness purposes, some connections between both communities have come about.

7.1. Convergence at the Modeling Step

We note that the link existing between formal techniques and simulation lies in the use of a model. This link is a good starting point to make them converge, even if their objectives are not the same. More specifically, the idea is to try to benefit from the formal performed work to establish the correctness regarding functional properties, and then to use this correct model to evaluate performance-related issues. Otherwise, another idea is to take advantage of the work performed at the performance level, by completing the existing and preliminary formal model with the performance model.

Some research to bridge both communities has already been carried out on the integration of nonformal and formal approaches in the case of validation of wired-based protocols. By surveying the literature, a lot of studies dealing with the transformation of functional models into a performance-oriented model can be found [Bütow et al. 1996; Heck 1996; Petriu and Woodside 2002; Mitschele-Thiel and Müller-Clostermann 1999; Steppler and Lott 1997], especially on the integration consisting of adding to the formal model nonfunctional properties. With such an approach, the work performed at a modeling step for validation purposes is reused for the performance evaluation. We do not need to perform the modeling work twice. As the purposes are not the same, the modeling step needs some adjustments to deal with performance constraints. It also needs to have an analysis method to solve the resultant model. Typically, the performance evaluation is performed after the functional design, when all the architectural decisions have been made.

The convergence of network research and protocol engineering research is becoming more and more necessary, especially with the new challenges that WSONs raise. The techniques are faced to their limits. In this way, we argue the importance of taking advantage of the differences between the techniques. This is also the observation made by Samper et al. [2006]. This article is, indeed, to our knowledge, the only one that applies formal models to present the different stack layers of a WSON node (in this case for the sensor networks) and their environment as well. The authors advocated using FDTs for all their advantages such as reliability, reusability, etc. (see Section 2). The basic formalism is the communicating parallel interpreted automata written in ReactiveML [Mandel and Pouzet 2005] and the hardware functionalities may be designed in VHDL (Very High-Speed Integrated Circuits Hardware Description Language). An interesting aspect is to formalize the environment, which may also be the cause of the different results obtained in simulation and real case studies. For that, the Lucky tool [Javier and Raymond 2005] was proposed. Finally, and like it is introduced in Samper et al. [2006], the convergence needs to occur not only at the modeling phase but also has to be considered for validation purposes.

7.2. Convergence at Validation Level

Although formal and nonformal approaches have their own main testing targets as well as their own characteristics and limitations, some attempts at convergence been carried out. The main goal is to get the advantage of their differences for covering as much as possible the phases of the protocol validation in its development lifecycle. But merging or blending both methods is not effortless. First, three entities are involved in this process: the two models provided by both communities plus the implementation under test. It means that the models have to be "equivalent," even if the design languages are quite different. Moreover, the system environments must be similar, which is not always simple regarding the WSON challenges and the limitations for both testing methods. It is also required that these three entities are correct and conform to each other, which is tough due to their different languages, viewpoints, power expressiveness, etc. Finally, and maybe one of the main problems is that both communities are very often distinct. Moreover, while clarifying some terms of vocabulary is needed, studies on eventual common languages, models, or tools are also necessary. We illustrate in the following some research attempts dealing with the mixing of both techniques.

The first step to orchestrate such bridge-building is to formalize the methodology for designing WSON protocols. Efforts have been achieved in that way by reimplementing the protocol stack layers of a simulated platform by utilizing formal refinement-based methodologies [Sgroi et al. 2000]. This enables the testers to perform architecture exploration much more quickly than using informal design techniques. Furthermore, the objective of this platform is to provide a entire product containing SW/HW. This formal design approach allows the correct partitioning between SW and HW, and hence prevents errors. Therefore, though this approach is not particularly dedicated to the validation of WSON protocols, it reveals that the design efforts for observing and/or controlling every functional and nonfunctional protocol aspects are necessary but costly.

By the way, research efforts have also attempted to cope with the language transformation mechanisms or their diverse and irrelevant expressiveness power. Fehnker et al. [2007] proposed a single top-level graphical model for simulation and model checking with the ability to easily collect, observe, and analyze the results. This kind of approach is mentioned as a "bridging-language" by the authors. In fact, it is currently quite relevant regarding their results and also in taking into account the applied interference model which has been verified and experimentally validated. Nevertheless, one of the issues, and not the least, is that a new simulator and dedicated specification language has to be used. This once again raises the difficulty for both communities of merging their efforts and adjusting their tools and languages accordingly.

Another interesting study was devoted to applying a formal testing approach merged with an emulated NS-2 platform [Maag et al. 2008]. Nevertheless, the specification confronted a huge number of topologies and, as previously mentioned, the generation of test sequences may lead to a state space explosion. The authors therefore applied the concept of nodes self-similarity, which allows mapping a route viewed by the implementation under test (IUT) for a given message in a real network with a path of the specification containing only three nodes. Hence, since the execution of test sequences (generated from this minimized formal model) on a real implementation is rather tough in life testing, the authors advocate for applying them through an emulated platform connected to a simulated mobile ad hoc network. A MANET routing protocol was tested in this manner. Notwithstanding, still many testing inconclusive verdicts have been provided. This is especially due to the unreliability of radio communications and because of the unexpected messages sent in the real network and not planned in the minimized model. This shows that even if we map formal models on an informal tool (such as a simulator/emulator), aspects concerning the other running protocols (such as 802.11) have to be taken into account through the formal approaches.

7.3. Ways for Converging

As mentioned before, nonfunctional aspects are mainly studied through performance-based tools and mathematical analysis. Nevertheless, formal approaches are more and more concerned about Quality-of-service (QoS) requirements and performances [McIver and Fehnker 2006; McIver 2006; Watteyne et al. 2005; Bhargavan et al. 2002b]. Unfortunately, it is clear that techniques for particular performance criteria for WSONs are still not tailored enough. Indeed, even if verification techniques may be applied to model checking, then the correctness of the requirements, the validation aspects are not reached, especially due to nonrealistic and incomplete formal models.

Based on this observation, it is interesting to reason backwards in studying works dealing with functional properties through nonformal approaches. Actually, this is met in almost all articles describing results obtained from simulators/emulators. Indeed, functional behaviors are implemented in those tools and performance results are based on these behaviors. That is the reason why the authors of Verisim [Bhargavan et al. 2002a] evaluated the correctness of the functional properties implemented in such performance-based tools. Verisim is a model combining NS-2 and the trace verification component provided by the *monitoring and checking* system, namely, MAC [Kim et al. 1999]. The goal is to generate an NS-2 trace T and to verify if the expected properties are included in the implementation I according to a scenario S. Finally, Kim et al. [1999] showed that the AODV implementation in NS-2 was false regarding some properties, for instance, the initial value for the hops number in a RREP. This work is very interesting and rather disturbing, raising several issues regarding the efficiency/reliability of simulation/emulation concerning the correctness of the functional behaviors.

But what does it mean exactly? Are all previous results obtained with NS-2/AODV false, and therefore not to be studied anymore? Actually not, because even if the performance results are noticeably distorted, a global understanding of the nonfunctional aspects are notwithstanding provided. Nevertheless, the most relevant meaning in that observation is the necessity of merging both formal and nonformal approaches to deliver the best results concerning both functional and nonfunctional WSON properties.

Nowadays, we may notice that while most of the functional properties are studied applying formal approaches, the nonfunctional ones are mainly tackled through nonformal techniques. And despite the increasing needs, especially with the WSONs becoming a trend and a matter of necessity in many domains, the melting idea is rarely met. “Rarely” because we may indeed cite one very promising work [Christmann et al. 2008]. In that article, the authors first presented a technique to associate model-driven development methods and a formal description technique (SDL) to design and specify an ad hoc protocol. But the very interesting step was the use of these first results to tackle model-driven performance simulations through NS-2. Indeed, the main result was to transform a complete SDL formal specification into an NS-2 executable model. However, even if that first article is relevant in a way for converging both worlds, much work is still to be done. Therefore, to cope with this needed blend, many targets may be pointed at. First, the languages used have to be commonly accepted by all communities. That is, it is nonsense studying different aspects of a same protocol (or worse, the same aspects) by using different languages. The languages should evolve (or be transformed from one phase to another) so they are able to take into account functional and nonfunctional properties. In this way, these languages could be applied to check the requirements (for verification), to generate test sequences (for testing), and to analyze the performance of a protocol. Thus, formal and nonformal techniques would be used harmoniously, providing common and real results.

8. CONCLUSIONS

Wireless self-organizing networks (WSONs) have attracted considerable attention from the network research community. Nevertheless, the success of WSON-related applications is strongly related to the well-done validation of properties of the network protocols involved. In this article, we have investigated the existing validation approaches and discussed their components and similarities. We have discussed the particularities introduced by multihop wireless networks and how to take advantage of similarities between the validation approaches to obtain complementary techniques. In summary, our goal was (1) to discuss the *foundations* of validation techniques and the difficulties imposed by WSONs characteristics, and (2) to give hints of open research problems.

ACKNOWLEDGMENTS

The authors wish to thank the anonymous referees whose insightful comments have significantly strengthened this article. We would like to also thank Dr. Artur Ziviani of the National Laboratory for Scientific Computing (LNCC) for fruitful discussions and useful remarks on the previous version of this work.

REFERENCES

ETSI. 2007. Methods for testing and specification (MTS); The testing and test control notation version 3; part 1: Ttcn-3 core language, v3.2.1. Tech. rep., 873-1, ETSI, Sophia Antipolis, France.

ISO. 1994. Information technology—open systems interconnection—conformance testing methodology and framework–part 1: General concepts. Tech. rep., 9646-1, ISO, Genewa, Switzerland.

Abdesslem, F. B., Iannone, L., M. D. de Amorim, K. O., Solis, I., and Fdida, S. 2007. A prototyping environment for wireless multihop networks. In *Proceedings of the Asian Internet Engineering Conference (AINTEC)*. 38–43.

Alur, R. and Dill, D. 1990. Automata for modeling real-time systems. In *Proceedings of the International Colloquium on Automata, Languages and Programming*. Vol. 443. 321–335.

Andel, T. R. and Yasinsac, A. 2006. On the credibility of manet simulations. *IEEE Comput. 39,* 7, 48–54.

Bagrodia, R. and Takai, M. 1999. Position paper on validation of network simulation models. In *Proceedings of the DARPA/NIST Network Simulation Validation Workshop.*

Balci, O. 1997. Verification, validation and accreditation of simulation models. In *Procedings of the 29th Conference on Winter Simulation*. 135–141.

Ballarini, P. and Miller, A. 2006. (Towards) model checking medium access control for sensor networks. In *Proceedings of the 2nd IEEE International Symposium on Leveraging Applications of Formal Methods*. 256–262.

Bayse, E., Cavalli, A., Núñez, M., and Zaïdi, F. 2005. A passive testing approach based on invariants: Application to the wap. *Computer Netw. 48*, 247–266.

Benbadis, F., de Amorim, M. D., and Fdida, S. 2005. Elip: Embedded location information protocol. In *Proceedings of the IFIP Networking Conference*.

Bhargavan, K., Gunter, C. A., Kim, M., Lee, I., Obradovic, D., Sokolsky, O., and Viswanathan, M. 2002a. Verisim: Formal analysis of network simulations. *IEEE Trans. Softw. Eng. 28,* 2, 129–145.

Bhargavan, K., Obradovic, D., and Gunter, C. A. 2002b. Formal verification of standards for distance vector routing protocols. *J. ACM 49,* 4, 538–576.

Bianchi, G. 2000. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE J. Select. Areas Comm. 18,* 3, 535–547.

Boehms, B. 1988. A spiral model of software development and enhancement. In *IEEE Comput. 21*. 61–72.

Bolch, G., Greiner, S., de Meer, H., and Trivedi, K. S. 2006. *Queueing Networks and Markov Chains*. John Wiley and Sons, New York, NY.

Boorstyn, R., Kershenbaum, A., Maglaris, B., and Sahin, V. 1987. Throughput analysis in multihop CSMA packet radio networks. *IEEE Trans. Comm. 35,* 3, 267–274.

Borujerdi, M. M. and Mirzababaei, S. M. 2004. Formal verification of a multicast protocol in mobile networks. *Int. J. Signal Proc. 1,* 4, 212–218.

Bütow, M., Mestern, M., Schapiro, C., and Kritzinger, P. S. 1996. Performance modelling with the formal specification language sdl. In *Proceedings of the 16th International Conference on Formal Description Techniques: Theory, application and tools*. Chapman & Hall, Ltd., London, U.K., 213–228.

Cavalli, A., Mederreg, A., Zaïdi, F., Combes, P., Monin, W., Castanet, R., MacKaya, M., and Laurenčot, P. 2004. A multi-services and multi-protocol validation platform—experimentation results. In *Proceedings of the 16th IFIP International Conference on Testing of Communication Systems*. Lectures Notes in Computer Science, vol. 2978, Springer, Berlin, Germany, 17–32.

Cavin, D., Sasson, Y., and Schiper, A. 2002. On the accuracy of MANET simulators. In *Proceedings of the 2nd ACM International Workshop on Principles of Mobile Computing* (POMC). 38–43.

Chakrabarti, A., Chatterjee, K., Henzinger, T., Kupferman, O., and Majumdar, R. 2005. Verifying quantitative properties using bound functions. In *Proceedings of Correct Hardware Design and Verification Methods (CHARME)*. 50–64.

Chaudet, C., Lassous, I. G., Thierry, E., and Gaujal, B. 2004. Study of the impact of asymmetry and carrier sense mechanism in ieee 802.11 multi-hops networks through a basic case. In *Proceedings of the 1st ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*. 1–7.

CHIYANGWA, S. AND KWIATKOWSKA, M. 2005. A timing analysis of AODV. In *Proceedings of the 7th IFIP WG 6.1 International Conference on Formal Methods for Open Object-Based Distributed Systems*. Lecture Notes in Computer Science, vol. 3535. Springer-Verlag, Berlin, Germany, 306–321.

CHRISTMANN, D., BECKER, P., GOTZHEIN, R., AND KUHN, T. 2008. Model-driven development of a MAC layer for ad-hoc networks with SDL. In *Proceedings of 1st Workshop on ITU System Design Languages*.

CLARKE, E. M. AND EMERSON, E. A. 1981. Synthesis of synchronization skeletons for branching time temporal logic. In *Proceedings of the 1st Workshop on Logic of Programs*.

CLARKE, E. M., GRUMBERG, O., AND PELED, D. A. 1999. *Model Checking*. MIT Press, Cambridge, MA.

CLAUSEN, T. AND JACQUET, P. 2003. RFC3626: *Optimized link state routing protocol (OLSR)*. www.ietf.org.

CORBETT, J. C., DWYER, M. B., HATCLIFF, J., LAUBACH, S., PUASUAREANU, C. S., ROBBY, AND ZHENG, H. 2000. Bandera: extracting finite-state models from Java source code. In *Proceedings of the International Conference on Software Engineering*. 439–448.

CURREN, D. 2005. A survey of simulation in sensor networks. Tech. rep. University of Binghamton, Binghamton, NY. www.cs.binghamton.edu/ kang/teaching/cs580s/david.pdf.

DA SILVA, A. P. C. 2006. Computational methods for markov reward models. Ph.D. dissertation, Federal University of Rio de JaneiroBonn, Rio de Janeiro, Brazil.

DA SILVA, A. P. C., LEÄO, R. M. M., AND DE SOUZA E SILVA, E. 2004. An efficient approximate technique for solving fluid models. *SIGMETRICS Perform. Eval. Rev. 32,* 2, 6–8.

DAS, S. AND DILL, D. L. 2002. Counter-example based predicate discovery in predicate abstraction. In *Proceedings of the 4th International Conference on Formal Methods in Computer-Aided Design*. Lecture Notes in Computer Science, vol. 2517. Springer-Verlag, Berlin, Germany, 19–32.

DE ALFARO, L., FAELLA, M., AND STOELINGA, M. I. A. 2004. Linear and branching metrics for quantitative transition systems. *Lecture Notes in Computer Science* vol. 3142. Springer-Verlag, Berlin, Germany, 97–109.

DE RENESSE, R. AND AGHVAMI, A. H. 2004. Formal verification of ad hoc routing protocols using spin model checker. In *Proceedings of the 12th Mediterranean Electrotechnical Conference*.

DE SOUZA E SILVA, E. AND GAIL, H. 2000. *Transient Solutions for Markov Chains*. In W. Grassmann, Ed., Kluwer, Dordrecht, The Netherlands.

DOUCET, A., DE FREITAS, N., AND GORDON, N. 2001. *Sequential Monte Carlo Methods in Practice*. John Wiley and Sons, New York, NY.

EISNER, C. 2005. Formal verification of software source code through semi-automatic modelling. *Softw. Syst. Model. 4,* 1, 14–31.

ELWALID, A. I. AND MITRA, D. 1992. Fluid models for the analysis and design of statistical multiplexing with loss priorities on multiple classes of bursty traffic. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies.*. 415–425.

EMERSON, E. A. 1990. *Temporal and Modal Logic*. MIT Press, Cambridge, MA.

ENGLER, D. AND MUSUVATHI, M. 2004. Static analysis versus software model checking for bug finding. In *Proceedings of the 5th International Conference on Verification, Model checking and Abstract Interpretation*. 191–210.

FEHNKER, A., FRUTH, M., AND MCIVER, A. 2007. Graphical modelling for simulation and formal analysis of wireless network protocols. In *Proceedings of the Workshop on Methods, Models and Tools for Fault Tolerance at the 7th International Conference on Integrated Formal Methods* (IFM). 80–87.

FEHNKER, A. AND GAO, P. 2006. Formal verification and simulation for performance analysis for probabilistic broadcast protocols. *Lecture Notes in Computer Science*, vol. 4104. Springer-Verlag, Berlin, Germany, 128–141.

FERNÁNDEZ-IGLESIAS, M. J., BURGUILLO-RIAL, J. C., F. J. GONZÁLEZ-CASTA N., AND LLAMAS-NISTAL, M. 2005. Wireless protocol testing and validation supported by formal methods: A hands-on report. *J. Syst. Softw. 75,* 1-2, 139–154.

GARETTO, M., GIACCONE, P., AND LEONARDI, E. 2007. Capacity scaling in delay tolerant networks with heterogeneous mobile nodes. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing.* 15–17.

GARETTO, M., GIACCONE, P., AND LEONARDI, E. 2008a. Capacity scaling of sparse mobile ad hoc networks. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies.* 206–210.

GARETTO, M., SALONIDIS, T., AND KNIGHTLY, E. 2006. Modeling per-flow throughput and capturing starvation in CSMA multi-hop wireless networks. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies*. 1–13.

GARETTO, M., SALONIDIS, T., AND KNIGHTLY, E. W. 2008b. Modeling per-flow throughput and capturing starvation in CSMA multi-hop wireless networks. *IEEE/ACM Trans. Netw. 16,* 4, 864–877.

Girod, L., Elson, J., Cerpa, A., Stathopoulos, T., Ramanathan, N., and Estrin, D. 2004a. Emstar: A software environment for developing and deploying wireless sensor networks. In *Proceedings of USENIX General Track*.

Girod, L., Stathopoulos, T., Ramanathan, N., Elson, J., Estrin, D., Osterweil, E., and Schoellhammer, T. 2004b. A system for simulation, emulation, and deployment of heterogeneous sensor networks. In *Proceedings of the ACM SIGOPS International Conference on Embedded Networked Sensor Systems*.

Glässer, U. and Gu, Q.-P. 2005. Formal description and analysis of a distributed location service for mobile ad hoc networks. *Theor. Comput. Sci. 336,* 2-3, 285–309.

Godary, K., Augé-Blum, I., and Mignotte, A. 2004. Sdl and timed Petri nets versus uppaal for the validation of embedded architecture in automative. In *Proceedings of the 1st Forum on Specification and Design Languages* (FDL).

Godefroid, P. 1997. Model checking for programming languages using Verisoft. In *Proceedings of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. ACM Press, New York, NY, 174–186.

Grossglauser, M. and Tse, D. 2002. Mobility increases the capacity of ad-hoc wireless networks. *IEEE/ACM Trans. Netw. 10,* 4, 477–486.

Gunter, E. and Peled, D. 2005. Model checking, testing and verification working together. *Form. Aspects Comput. 17,* 2, 201–221.

Hansson, H. and Jonsson, B. 1994. A logic for reasoning about time and reliability. *Form. Aspects Comput. 6,* 5, 512–535.

Havelund, K. and Pressburger, T. 2000. Model checking Java programs using Java pathfinder. *Softw. Tools Tech. Trans. 2,* 4, 366–381.

Heck, E. 1996. Performance evaluation of formally specified systems—the intergration of SDL with hit. Ph.D. dissertation, University of Dortmund, Dortmund, Germany.

Hoffman, L. 2008. In search of dependable design. *Comm. ACM 51,* 7, 14–16.

HOL. 2005. `http://www.cl.cam.ac.uk/Research/HVG/HOL/`.

Holzmann, G. J. 1991. *Design and Validation of Computer Networks*. Prentice-Hall International Editions, Englewood Cliffs, NJ.

Holzmann, G. J. 2003. *The SPIN Model Checker*. Addison-Wesley, Reading, MA.

ITU-T 1996. *Message Sequence Charts (MSC'96)*. ITU-T, Geneva, Switzerland.

Jacobson, I., Booch, G., and Rumbaugh, J. 1999. *The Unified Software Development Process*. Addison-Wesley, Reading, MA.

Javier, E. and Raymond, P. 2005. The lucky language reference manual. Tech. rep. TR-2004-06, Verimag, Giéres, France.

Jeffries, R. and Ambler, S. W. 2002. *Agile Modeling*. Jon Wiley & Sons, New York, NY.

Jones, V., Rensink, A., Ruys, T., Brinksma, E., and van Halteren, A. 2004. A formal MDA approach for mobile health systems. In *Proceedings of the 2nd European Workshop on Model Driven Architecture with an emphasis on Methodologies and Transformations*. 28–35.

Keshav, S. 1988. REAL: A network simulator. Tech. rep. 88/472. University of California, Berkeley, Berkeley, CA.

Kiess, W. and Mauvea, M. 2007. A survey on real-world implementations of mobile ad-hoc networks. *Ad Hoc Netw. 5,* 3, 538–576.

Kim, M., Viswanathan, M., Ben-Abdallah, H., Kannan, S., Lee, I., and Sokolsky, O. 1999. Formally specified monitoring of temporal properties. In *Proceedings of the 11th Euromicro Conference on Real-Time Systems*. 1–14.

Kindler, E. 1994. Safety and liveness properties: A survey. *Bull. Eur. Assoc. Theor. Comput. Sci.* 53, 268–272.

Kleinrock, L. 1975. *Queueing Systems*. Wiley-Interscience, New York, NY.

Kropff, M., Krop, T., Hollick, M., Mogre, P. S., and Steinmetz, R. 2006. A survey on real world and emulation testbeds for mobile ad hoc networks. In *Proceedings of the 2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities* (TRIDENTCOM).

Kurkowski, S., Camp, T., and Colagrosso, M. 2005. MANET simulation studies: the incredibles. *ACM SIGMOBILE Mob. Comput. Comm. Rev. 9,* 4, 50–61.

Kwiatkowska, M., Norman, G., and Parker, D. 2005. Probabilistic model checking and power-aware computing. In *In Proceedings of the 7th International Workshop on Performability Modeling of Computer and Communication Systems* (PMCCS).

Kwiatkowska, M., Norman, G., Segala, R., and Sproston, J. 2002a. Automatic verification of real-time systems with discrete probability distributions. *Theor. Comput. Sci. 282*, 101–150.

KWIATKOWSKA, M., NORMAN, G., AND SPROSTON, J. 2002b. Probabilitic model checking of the ieee 802.11 wireless local area network protocol. In *Proceedings of the 2nd Joint International Workshop on Process Algebra and Probabilitics Methods, Performance Modeling and Verification*. 169–187.

KWON, S. M. AND KIM, J. S. 2008. Coverage ratio in the wireless sensor networks using monte carlo simulation. In *Proceedings of the 4th International Conference on Networked Computing and Advanced Information Management*. 235–238.

LAHIRI, S. K. 2004. Unbounded system verification using decision procedure and predicate abstraction. Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA.

LAHIRI, S. K. AND BRYANT, R. E. 2005. UCLID-PA homepage: Verification tool for UCLID models using predicate abstraction. Carnegie Mellon University, Pittsburgh, PA.

LAHIRI, S. K. AND BRYANT, R. E. 2007. Predicate abstraction with indexed predicates. *ACM Trans. Computat. Log. 9,* 1, 4.

LARSEN, K. G., PETTERSON, P., AND YI, W. 1997. Uppaal in a nutshell. *Int. J. Softw. Tools Techn. Trans. 1,* 2, 134–152.

LEAO, R. M. M., DE SOUZA E SILVA, E., AND DINIZ, M. C. 2001. Traffic engineering using reward models. In *Proceedings of the International Teletraffic Congress*. 1101–1112.

LIN, T., MIDKIFF, S., AND PARK, J. 2003. A framework for wireless ad hoc routing protocols. In *Proceedings of the IEEE Wireless Communications and Networking Conf.* (WCNC). 1162–1167.

LUNDGREN, H., NORDSTRÖM, E., AND TSCHUDIN, C. 2002. Coping with communication gray zones in ieee 802.11b based ad hoc networks. In *Proceedings of the 5th ACM international workshop on Wireless Mobile Multimedia*. 49–55.

MAAG, S., GREPET, C., AND CAVALLI, A. 2008. A formal validation methodology for manet routing protocols based on nodes' self similarity. *Comp. Comm. 31,* 4, 827–841.

MAAG, S. AND ZAIDI, F. 2006. Testing methodology for an ad hoc routing protocol. In *Proceedings of the ACM International Workshop on Performance Monitoring, Measurement, and Evaluation of Heterogeneous Wireless and Wired Networks*. 48–55.

MAAG, S. AND ZAIDI, F. 2007. A step-wise validation approach for a wireless routing protocol. *Posts, Telecomm. Inform. Tech. J. 1*, 34–40.

MACKER, J. P., CHAO, W., AND WESTON, J. W. 2003. A low-cost, IP-based mobile network emulator (MNE). Tech. rep. ADA464904. Naval Research Lab, Washington, D.C.

MANDEL, L. AND POUZET, M. 2005. ReactiveML, a reactive extension to ML. In *Proceedings of the 7th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming* (PPDP).

MANNA, Z., BJØRNER, N. S., BROWNE, A., COLÓN, M., FINKBEINER, B., PICHORA, M., SIPMA, H. B., AND URIBE, T. E. 1999. An update on STeP: Deductive-algorithmic verification of reactive systems. In *Tool Support for System Specification, Development and Verification*, R. Berghammer and Y. Lakhnech, Eds. Advances in Computing Science. Springer-Verlag, Berlin, Germany, 174–188.

MARKOV, A. 1971. *Extension of the Limit Theorems of Probability Theory to a Sum of Variables Connected in a Chain. Markov Chains*. John Wiley and Sons, New York, NY.

MARTIN, J. 1990. *RAD, Rapid Application Development*. MacMillan Publishing Co, New York, NY.

MCIVER, A. 2006. Quantitative mu-calculus analysis of power management in wireless networks. In *Proceedings of the 3rd International Theoretical Aspects of Computing*. 50–64.

MCIVER, A. K. AND FEHNKER, A. 2006. Formal techniques for the analysis of wireless networks. In *Proceedings of the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*. 263–270.

MEROUANE, K., GREPET, C., AND MAAG, S. 2007. A methodology for interoperability testing of a manet routing protocol. In *Proceedings of the 3rd International Conference on Wireless and Mobile Communications*. IEEE Computer Society Press, Los Alamitos, CA.

MITSCHELE-THIEL, A. AND MÜLLER-CLOSTERMANN, B. 1999. Performance engineering of sdl/msc systems. *Comp. Netw. ISDN Syst. 31,* 17, 1801–1816.

PERKINS, C., BELDING-ROYER, E., AND DAS, S. 2003. Rfc 3561: ad hoc on-demand distance vector (AODV) routing. IETF Tech. rep. `www.ietf.org`.

PETERSON, J. L. 1977. Petri nets. *ACM Comp. Surv. 9,* 3, 223–252.

PETERSON, J. L. 1981. *Petri Net Theory and the Modeling of Systems*. Prentice Hall, Englewood Cliffs, NJ.

PETRI, C. A. 1962. Kommunikation mit automaten. Ph.D. dissertation, University of Bonn, Bonn, Germany.

PETRIU, D. AND WOODSIDE, M. 2002. Software performance models from systems scenarios in use case maps. In *Proceedings of the 12th International Conference on Performance Evaluation: Modelling Techniques and Tools*. Springer Verlag, Berlin, Germany, 1–8.

PIZZONIA, M. AND RIMONDINI, M. 2008. Easy emulation of complex networks on inexpensive hardware. In *Proceedings of the 4th International Conf. on Testbeds and Research Infrastructures for the Development of Networks and Communities* (TRIDENTCOM).

PLATEAU, B. AND ATIF, K. 1991. Stockastic automata network of modeling parallel systems. *IEEE Trans. Softw. Eng. 17,* 10, 1093–1108.

PNUELI, A. 1981. A temporal logic of programs. *Theor. Comp. Sci. 13,* 1, 45–60.

PUZAR, M. AND PLAGEMANN, T. 2005. NEMAN: A network emulator for mobile ad-hoc networks. Tech. rep. TR321. Department of Informatcis, University of Oslo, Oslo, Norway.

RAY, S., STAROBINSKI, D., AND CARRUTHERS, J. B. 2005. Performance of wireless networks with hidden nodes: A queuing-theoretic analysis. *J. Comp. Comm. (Special Issue on the Performance Issues of Wireless LANs, PANs and Ad Hoc Networks. 28,* 10, 1179–1192.

RIMONDINI, M. 2007. Emulation of computer networks with Netkit. Tech. rep. TR RT-DIA-113-2007. University of Roma Tre, Rome, Italy.

ROYCE, W. W. 1987. Managing the development of large software systems: concepts and techniques. In *Proceedings of the International Conference on Software Engineering Table of Contents*. 328–338.

RUBINSTEIN, R. Y. AND KROESE, D. P. 2007. *Simulation and the Monte Carlo Method*. John Wiley & Sons, New York, NY.

RUTTEN, J., M.KWIATKOWSKA, NORMAN, G., AND PARKER, D. 2004. Mathematical techniques for analysing concurrent and probabilitics systems. *CRM Monograph Series 23*. American Mathematical Society, Providence, RI.

SAMPER, L., MARANINCHI, F., MOUNIER, L., AND MANDEL, L. 2006. Glonemo: Global and accurate formal models for the analysis of ad-hoc sensor networks. In *Proceedings of the 1st International Conference on Integrated Internet Ad hoc and Sensor Networks*.

SGROI, M., DA SILVA, J. L., BERNARDINIS, F. D., BURGHARDT, F., SANGIOVANNI-VINCENTELLI, A., AND RABAEY, J. 2000. Designing wireless protocols: Methodology and applications. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. 3726–3729.

SPIVEY, J. M. AND SPIVEY, M. 1996. *An Introduction to Logic Programming Through Prolog*. Prentice Hall, Englewood Cliffs, NJ.

STEPPLER, M. AND LOTT, M. 1997. Speet—SDL performance evaluation tool. In *Proceedings of SDL'97*. 53–67.

STERLING, L. AND SHAPIRO, E. 1994. *The Art of Prolog*. MIT Press, Cambridge, MA.

TAN, C. L. AND PINK, S. 2000. Mobicast: a multicast scheme for wireless networks. *Mobilz Netw. Appl. 5,* 4, 259–271.

TICKOO, O. AND SIKDAR, B. 2004. Queueing analysis and delay mitigation in IEEE 802.11 random access MAC based wireless networks. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies*. 7–11.

VAHDAT, A. AND BECKER, D. 2000. Epidemic routing for partially-connected ad hoc networks. Tech. rep. Duke University, Durham, NC.

VERILOG. 1997. *ObjectGEODE Simulator*.

VLIET, H. V. 1999. *Sotware Engineering: Principles and Practice (2nd Edition)*. Wiley, New York, NY.

WANG, Y. AND XIONG, M. 2005. Monte Carlo simulation of leach protocol for wireless sensor networks. In *Proceedings of the 6th International Conference on Parallel and Distributed Computing Applications and Technologies*. 85–88.

WATTEYNE, T., AUGE-BLUM, I., AND UBEDA, S. 2005. Formal QoS validation approach on a real-time MAC protocol for wireless sensor networks. Tech. Rep. 5782, RR-5782. INRIA, Rocquencourt, France.

WEI, M., DUBOIS, F., VINCENT, D., AND COMBES, P. 2003. Looking for better integration of design and performance engineering. In *Proceedings of SDL 2003: System Design*, Springer, Berlin, Germany, 1–17.

WIBLING, O. 2005. Ad hoc routing protocol validation. M.S. thesis. Uppsala University, Uppsala, Sweden.

XU, N., RANGWALA, S., CHINTALAPUDI, K., GANESAN, D., BROAD, A., GOVINDAN, R., AND ESTRIN, D. 2004. A wireless sensor network for structural monitoring. In *Proceedings of the ACM SIGOPS International Conference on Embedded Networked Sensor Systems*. 13–24.

ZAÏDI, F., CAVALLI, A., AND BAYSE, E. 2009. Network protocol interoperability testing based on contextual. In *Proceedings of the 24th Annual ACM Symposium on Applied Computing*. 321–327. TO be published.

ZAKKUIDIN, I., HAWKINS, T., AND MOFFAT, N. 2005. Towards a game theoretic understanding of ad hoc routing. *Electron. Notes Theor. Comp. Sci. 119,* 1, 67–92.

Received July 2008; revised January 2009, March 2009; accepted April 2009