An Overview of MANETs Simulation

Luc Hogie [1,4]

Université du Luxembourg
Luxembourg

Pascal Bouvry [2]

Université du Luxembourg
Luxembourg

Frédéric Guinand [3]

Laboratoire d'Informatique
Université du Havre
France

Abstract

Mobile Ad hoc NETworks (MANETs) are dynamic networks populated by mobile stations. Stations in MANETs are usually laptops, PDAs or mobile phones. These devices feature Bluetooth and/or IEEE 802.11 (WiFi) network interfaces and communicate in a decentralized manner. Mobility is a key feature of MANETs. Because of their high cost and their lack of flexibility of such networks, experimentation is mostly achievable through simulation. Numerous tools exist for MANETs simulation, including *ns-2* and GloMoSim which are the two most popular ones. This paper provides a State of the Art of MANETs simulators and associated simulation techniques. First it gives an overview of the domain. Then it provides a map of the main characteristics that MANETs simulation tools should feature and the current support of these. Finally, a description for each simulator is provided, including an explanation of what make them appealing solutions.

Key words: Mobile ad hoc networks, MANETs, simulation.

[1] Email: luc.hogie@uni.lu
[2] Email: pascal.bouvry@uni.lu
[3] Email: frederic.guinand@univ-lehavre.fr
[4] Thanks to Steffen Rothkugel. Email: steffen.rothkugel@uni.lu

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

1 Introduction

Mobile ad hoc networks (MANETs) are networks composed of a set of communicating devices able to spontaneously interconnect without any pre-existing infrastructure. Devices in range can communicate in a point-to-point fashion. In addition to that, these devices are generally mobile.

More and more people are interested in ad hoc networks. Not only their importance in military applications is growing, but also their impact on business is increasing. The wide spread of lightweight and low-cost mobile devices—we are talking about mobile phones, PDAs, Pocket PCs, etc—which now embed Bluetooth and WiFi (IEEE 802.11) network adapters enable the spontaneous creation of city-wide MANETs. These networks could then constitute the infrastructure of numerous applications such as emergency and health-care systems [44], groupware [18], gaming [61][31][57], advertisements, customer-to-customer applications (like the UbiBay project [30]), etc.

Investigating MANETs is achievable by resorting either to software-based simulators or to experimentation networks (testbeds). Most researchers favour simulators as the expense of testbeds. What prevents (or at least hinders) the use of real-size testbeds is their cost and their inherent lack of flexibility. This becomes particularly impeding as the size of the experimented network grows. Software-based simulation then turns out to be a viable alternative and a widely used solution. This article surveys MANETs simulators. It is organized as follows: In section 2 testbeds solutions are overviewed, although they do not constitute the focus of this paper. Next in section 3, crucial aspects of MANETs simulation are exposed. The techniques employed to implement them are described. Then in section 4, a list of the documented simulators is provided. Finally, section 5 gives some hints on which simulator to use for what needs and section 6 concludes by summarizing the current trends in MANETs simulation and by foreseeing its future directions.

Please note that this paper does not survey wired network simulators [5] and sensor network simulators [62][54]. The reason is that wired and sensor networks considerably diverge from MANETs in terms of structure, technologies, applications, etc. Thus they are considered to be out of the topic tackled herein.

2 Testbeds

Testbeds are in-lab networks built and used by researchers. They aim at enabling the experimentation of protocols and applications. A survey of existing testbeds is proposed by De and al. [25]. Among all existing testbeds, let us highlight the following ones:

- The APE project [48] experimented routing on 37 IEEE 802.11 nodes. Numerous participants were involved—particularly for moving the devices—which impacted the high costs and difficulty of the experiments.

- The RoofNet project [24], conducted by researchers at MIT over the city of Cambridge (UK), uses 40 IEEE802.11 nodes. It primarily aims at studying mesh networks but it is also employed at investigating multi-hop routing on MANETs. Citizens can connect to the Internet through the research network. The RoofNet project uses static nodes. It hence does not deal with mobility issues.
- Focusing on multi-hop routing, Douglas and Raichle [42] used a MANET testbed made of 8 IEEE 802.11 nodes.
- Ritter and al. [61] built a Bluetooth network for experimenting gaming and home automation. In order to have a global view of the network—which makes monitoring it a lot easier—they used 300m-range EWM adapters.
- The Ad Hoc City project [40] uses an mixed approach: both wired and wireless stations are used. By covering in time and space the whole city of Seattle, real information of nodes' movements could be gathered. This testifies the current trend towards realistic mobility models.

Testbeds suffer from several drawbacks. More precisely, the cost of the hardware (one node is several hundred euros) coupled with the difficulty of managing applications—in terms of deployment, monitoring, etc—over such testbeds makes that only a few testbeds could be built up to now. In order to overcome these limitations of real-scale testbeds, assorted solutions were proposed. Among them:

- the EWANT project [63] reduces the dimensions of the network to make it fit on a table-top. The downscaling of the radio links is achieved by the use of two-way attenuators.
- Zhang and Li [75] emulate the IEEE802.11b communication links by ethernet cables. The impact of mobility is replicated by altering the routing tables.

These techniques simplify the use of testbeds, but they in no way improve their scalability. In other words, they do not permit the construction of bigger networks. In the analysed literature no testbed of more than 50 nodes were proposed. The fact that no large testbed is currently available or could easily be built is a major problem for validating new concepts/applications for MANETs. Indeed many studies drastically require the use of large networks, as explained by Riley and Ammar [58].

3 MANETs simulation techniques

None of the defects of testbeds (see section 2) is applicable to simulators. More precisely, as simulators allow the network to be handled as a whole, they are way more handy to use and they make the monitoring easy. Moreover, as experimentations are described as scenarios files, they are reproducible. Besides, the size of the simulated network is upperbounded only by the computational

power available (we will see that only a few of the simulators can address large scale problems).

Because of the complex nature of the MANETs, their simulation is a very challenging issue. Simulators rely on various techniques for improving their accuracy, speed, scalability, usability, etc. This section gives an overview of the challenges faced and of the strategies employed.

3.1 The accuracy of MANETs simulators

There have been some studies [58][32][21] focusing on the accuracy of simulations. Some of them have pointed out that there exist some significant variations in the way simulators operate. One cannot state that these variations can be expressed in terms of accuracy. Formally speaking, no network simulator is accurate. At best a simulator can be said to be *dependable* and *realistic*. Researchers who drastically need accuracy will want to conduct their experiments on the real devices, using testbeds. When this is not possible they will have to resort to simulation and hence to content with a certain level of imprecision. Imprecision has various causes, explained in the following.

3.1.1 The impact of granularity

Software architects know that building a computerized model that include all details of the targeted domain is merely impossible: computerized models of the real-world are necessarily designed with a certain granularity. Indeed, modelling everything—up to the electrons and atoms—is not achievable. In the specific case of MANETs modelling, if software layers are relatively easy to re-implement within simulators, modelling the hardware inevitably leads to severe compromises.

Unfortunately, as studied by Heidemann and al. [32], neglecting details has in some cases a serious impact on the result obtained. Ideally the granularity of the model used should be defined according to the needs of the simulated application.

Table 1 gives some elements of the dependability for each simulator. Concerning table 1, as no metrics is available for characterizing the level of granularity, we define `finest < finer < fine < medium < application-level`.

3.1.2 Mobility models

So far, most studies have made fairly unrealistic assumptions concerning node mobility. More precisely, they generally rely on randomized mobility models [13][71][41][52], especially on the Random Waypoint mobility model [72]. Thanks to several studies focusing on the wallop of randomized mobility [72][14][65], researchers are now aware of their harmful impact. Since then an effort towards more realistic mobility models can be observed through papers and projects like the Group Mobility Model [35], the Graph-based Mobility Model [67], the Obstacle Mobility Model [39][38], the UDEL model [15][16]

Name	Granularity	Metropolitan mobility
ns-2	Finest	Support
DIANEmu	Application-level	No
Glomosim	Fine	Support
GTNets	Fine	No
J-Sim	Fine	Support
Jane	Application-level	Native
NAB	Medium	Native
OMNet++	Medium	No
OPNet	Fine	Support
QualNet	Finer	Support
SWANS	Medium	-

Table 1
Elements of dependability: ganularity and mobility.

and the GEMM project [56]. Mobility models have been surveyed by Camp and al. [19] and Ray and Suprio [56].

3.1.3 Radio propagation models

Radio waves propagation constitutes another important aspect of dependability. Up to now, most studies have considered the free path loss propagation model, often coupled with randomized mobility patterns. Radio wave propagation is generally associated to mobility because both are constrained by the same environmental elements. Particularly, radio waves are subject to diffraction, refraction, and scattering. Up to now, no simulator implement these three properties of radio propagation. Current implementations rely either on statistical models [20] or on partial models of refraction. For instance, the UDel project [15][16] as well as Dricot and De Doncker [27] model radio waves by using ray-tracing and Markov-chains. Unfortunately these time-consuming techniques dramatically decelerate simulations (up to 100 times, as stated in [27]).

3.1.4 Simulation size

Riley and Ammar [58] explained that there exists a threshold of the number of stations in the network for which the results obtained no longer vary as the number of stations increases. This threshold depends on the simulated application. Because of the lack of scalability of most simulators, this impor-

tant aspect of protocol validation is generally neglected. Scalable simulators [37][73] and runtime improvements techniques [60][70][50] should overcome this problem.

3.2 Simulation acceleration techniques

Simulation can be either *continuous* or *discrete*. Continuous simulation makes use of analytic models. Because of the intrinsic complexity of the MANETs, analytic models can hardly be applied. Discrete simulation proves to be more practicable. In the case of MANETs simulation, discrete simulation can benefit from specific optimization techniques. This section gives an overview of general and such MANETs-specific techniques implemented within simulators.

Table 2 details the runtime properties for each simulator.

3.2.1 Parallelism and distribution

One the one hand, *parallelism* refers to the simultaneous execution of different instructions of the same program. It is used to quicken simulations. Parallelism is a relevant technology for simulating wired networks. One the other hand, *Distribution* refers to the repartition of program data or code (or both) on distinct computers. It is primarily used to bring scalability and/or to enable parallelism. Parallelism and distribution can be coupled or used independently. The improvement in terms of quickness/scalability depends on the number of processors/nodes involved. When distribution is not used, parallelism is typically applied on shared memory architectures (SMP). This technique allows simulators to model networks made of tens of thousands stations. It is implemented by GloMoSim [73]. Distribution benefits from the recent interest in beowulf clusters (local computational grids made of numerous low-cost workstations, typically PC/Linux boxes). *pdns* relies on this technique which allow it to simulate hundreds of thousands stations.

Let us point out the COMPASS [2] project at GeorgiaTech, which makes an extensive use of distribution by building heterogeneous distributed simulations consisting of instances of *ns-2* and GloMoSim operating together.

3.2.2 Staged simulation

Staged simulation is described by Walsh and Sirer [70]. It is a general technique which improves the performance of discrete-event simulators by identifying and eliminating redundant computations. It consists of three parts: function caching, event-restructuring, and time-shifting. *Function caching* avoids redundant computations by placing into a memory cache the arguments and results of function calls. *Event-restructuring* improves on function caching by exposing low-level events that otherwise would have not been treated by function caching. *Time-shifting* reorders the events into a sequence that is better suited to the computer architecture that executes the simulator. It also enables a sequence of small, consecutive events to be computed altogether

Name	Parallelism	Interface
ns-2	No	C++/OTCL
DIANEmu	No	Java
Glomosim	SMP/beowulf	Parsec (C-based)
GTNets	SMP/beowulf	C++
J-Sim	RMI-based	Java
Jane	No	Java
NAB	No	OCaml
OMNet++	MPI/PVM	C++
OPNet	Yes	C
pdns	beowulf	C++/OTCL
QualNet	SMP/beowulf	Parsec (C-based)
SWANS	No	Java

Table 2
How simulators are parallelized and how they can be programmed.

by a single, more efficient algorithm. The SNS project [70] applies staged simulation to *ns-2* and boasts a runtime 30 times faster than the original one.

3.2.3 Bining

Bining makes good use of the spatial localization of network nodes in the MANETs—it is actually practicable and widely employed in all systems made of spatially located objects. It consists in the division of the simulation area into a list-based (applicable on 1-dimensional simulation space), grid-based (also referred to as *flat bining*, applicable on 2D spaces) or tree-based (also referred to as *flat bining*, applicable on 3D spaces) structure. Bining dramatically improves the determination of the communication links in the network. For example, applying flat bining reduces the complexity of this process from $O(n^2)$ to $O(n)$ (n being the number of stations in the network). This technique is described by Naoumov and Gross [50], who applied it to *ns-2*.

3.2.4 Hybrid simulations

The principle of hybrid simulations is to mix analytic models and discrete ones. Although pure analytic models [64] do not suit MANETs simulation, mixing them with discrete model leads to good results. Hybrid simulation is being investigated and applied to MANETs simulation by Lu and Schormans [47].

3.3 Simulation languages and frameworks

Several simulators like GloMoSim [73] and SWANS [10] have been developed using languages, libraries and frameworks dedicated to discrete-event simulation. These middleware technologies typically focus on performance, concurrency and distribution. As detailed by Barr and al. [11], one approach has been to create new simulation languages that are closely related to popular existing languages, with extensions for message dispatch, synchronization and time management. Csim [66], Yaddes [55], Maisie [7], and Parsec [8], for example, are derivatives of C and C++. Others, such as Apostle [17] and TeD [53] have taken a more domain-specific language approach. Finally, projects like Moose [69], Sim++ [6], JIST [10], J-Sim [22] and Pool [4] investigated various object-oriented (OO) possibilities for check-pointing, inheritance, concurrency and synchronization in the context of simulation. Currently, the most popular development language/platform for MANETs simulation is Java McNab and Howell [49] discusses the pros and cons of Java for discrete-event simulation.

3.4 Visualization and debugging facilities

. It has been recognized that distributed programming is inherently difficult. In addition to being distributed, "Ad hoc applications" (a cutting-edge application class which define the application running on ad hoc networks) are typically decentralized. Worse, the highly dynamic nature of mobile ad hoc networks makes the development of protocols and applications extremely error-prone. It is then of a chief importance that the user is provided with efficient debugging and visualization mechanisms.

There exist two techniques which provide feedback on what happens within the simulation. Discrete-event simulation encourages the generation of a trace file containing a description a each event that occurred. This general technique reports all events to the user. Unfortunately because of the huge size of the generated trace, this technique consumes a lot of CPU resource and thus significantly slows down the simulation process. Finally, the user needs to write text-processing scripts (called *filters* in the Unix world) in order to retain only what he is interested in and to generate GNUPlot files. The alternative consists in dynamically interacting with the simulation engine by resorting to the *observer design pattern*. The measurement system is then event-driven. More precisely, the user initially announces the classes of events he is interested in. He will then be dynamically notified of such events as the simulation process progresses. DIANEmu [43] and GloMoSim [73] use such a technique. This saves a lot of CPU resource. Besides, it permits the construction of interactive graphical interfaces, which proves to be of a prime importance for debugging and monitoring purposes.

4 MANET simulators currently in use

The literature mentions less than twenty MANETs simulators currently in use. Note that wired network simulators [5] and sensor network simulators [62][54] are not taken into consideration.

Since the wireless extension for *ns-2* which constitutes the first MANETs simulator, numerous tools have been made available to the community. Some of them have even considerably broken through and are now massively used. Because of the variable needs of research projects, many researchers do not wish to use these simulators. Indeed not all research project focus on the lowest layers of the network stack. More and more people are looking at the highest layers, i.e. at developing new concepts and applications for MANETs (service discovery, customer-to-customer applications, gaming, etc). For example, Hellbrück and Fischer developed ANSim [33], an interactive MANETs simulator, in order to analyze the structural properties of the MANETs. Görgen and al. [31] work on ad hoc gaming, using the Jane simulator [29][46]. Working on advanced broadcasting protocols and messaging applications, Hogie and al. wrote Madhoc [34] because none of the simulators available both featured an interactive mode making debugging of broadcasting protocols easy and permitted the simulation of large networks. In order to define the Group Mobility Model, Hong and al. [35] had recourse to the Maisie language [7] in order to develop their custom simulator. More custom simulators are described in the following.

The simulators described in this section are either commercial solutions or lab-tools that broke through thanks to their qualities.

Table 3 provides an estimation of the popularity of the simulator and on the license they use. As there is no statistics available on simulators users, we consider the number of web-pages that refer to each simulator. This gives an order of magnitude of their popularity.

DIANEmu [43] is a discrete-event simulator developed at Karlsruhe University (Germany). It aims to enable the simulation of ad hoc applications in realistic contexts. So far, most simulators have been designed to permit simulations at a protocol-level. DIANEmu's approach is different: it assumes that the lowest network layers (up to the fourth one) are available. DIANEmu then focuses on the application model. DIANEmu belongs to a new class of simulators which allow the large-scale simulation of high-level applications such as gaming and e-business.

DIANEmu provides a complete environment for application design. Its simulation engine is closely coupled to its graphical interface. Attesting of its modern design, its measurement system is event-driven. More precisely it defines that to each event class is associated to a given handler (referred to as a *gauge*). This handler is then dynamically invoked when the events of the specified class occur. This technique is detailed in section 3.4. DIANEmu is written in Java and is free.

Name	Popularity	Licence
ns-2	88.8%	Open source
GloMoSim	4%	Open source
OPNet	2.61%	Commercial
QualNet	2.49%	Commercial
OMNet++	1.04%	Free for academic and educational use
NAB	0.48%	Open source
J-Sim	0.45	Open source
SWANS	0.3%	Open source
GTNets	0.13	Open source
pdns	$< 0.1\%$	Open source
DIANEmu	$< 0.1\%$	Free
Jane	$< 0.1\%$	Free

Table 3

GloMoSim [73] is developed at UCLA (California, USA). It is the second most popular wireless network simulator. GloMoSim is written in Parsec [8] and hence benefits from the latter's ability to run on shared-memory symmetric processor (SMP) computers. New protocols and modules for GloMoSim must be written in Parsec too. GloMoSim respects the OSI standard.

The parallelization technique used by GloMoSim is the same than *pdns*'s one; that is the network is split in different subnetworks, each of them being simulated by distinct processors. The network is partitioned in such a way that the number of nodes simulated by each partition is homogeneous.

GloMoSim uses a message-based approach to discrete-event simulation. More presicely, network layers are represented as objects called *entities*. Events are represented as time-stamped messages handled by entities. GloMoSim's network model does not define every network nodes as entities because this would lead to too numerous objects. Instead, GloMoSim uses entities to model network layers. Messages—which represent network events—then cross the layer stack by being interchanged by the entities. GloMoSim can simulate networks made of tens of thousands devices.

Just like *ns-2*, realistic simulation have been made possible by exten-

sions such as the Obstacle mobility model [39][38] and the GEMM project [56]. A java-based visualization tool is provided.

The qualities of GloMoSim permitted it to be chosen as the core of the commercial QualNet simulator (detailed hereinafter). Although it is a good tool for MANETs simulation, GloMoSim suffers from a lack of a good and in-depth documentation.

GTNets GTNets [59] is developed at GeorgiaTech institute (Atlanta, USA). According to its authors, the design philosophy of GTNetS is to create a simulation environment that is structured much like actual networks are structured. More precisely, in GTNetS, there is clear and distinct separation of protocol stack layers and the network programming interface used by applications use function calls similar to the ubiquitous POSIX standard. The parallelization ability of GTNetS makes it possible to distribute a single simulation over either a network of loosely coupled workstations, a shared-memory symmetric multiprocessing system (SMP), or a combination of both. This endows GTNetS with good scalability and then allows the simulation of large networks. Concerning the support of protocols, IEEE 802.11 as well as Bluetooth [74] are implemented. Another benefit of GTNets is that the simulator gathers statistics regarding its own performance. The graphical user interface provided with GTNetS supports the graphical representation of the simulation topology, with selective enabling and disabling of display for specified nodes and links. It is open source.

J-Sim [22] (formerly known as JavaSim) is developed at Ohio and Illinois Universities. It is a component-based, compositional simulation environment. Initially designed for wired network simulation, its Wireless Extension proposes an implementation of the IEEE 802.11 MAC—which is the only MAC supported so far. This extension turns J-Sim to a viable MANETs simulator. J-Sim also features a set of components which facilitates basic studies of wireless/mobile networks, including three distinct radio propagation models and two stochastic mobility models. J-Sim is written in Java and is open source.

Jane [29][46] is developed at Trier University (Germany). It consists of both a simulation environment and an execution platform. Its main interestingness is that it allows the simulation code to be migrated to the real devices without any modification. Jane also features an emulation mode that allows real devices to participate to simulations. In addition to that, Jane features high-level concepts (such as the notions of *service*, *message*, etc) that are suitable to the simulation of applications-level services. It also makes use of GPS information, what turns it to an appealing tool for the simulation location-based services. Jane is written in Java and is open source.

NAB [37] (Network in A Box) is a discrete event simulator developed at EPFL

(Lausanne, Switzerland). NAB is dedicated to MANETs simulation. By focusing on scalability and visualization and by featuring a very realistic mobility model (a constrained waypoint based on city maps), it meets the needs of cutting-edge applications. According to its author, Nab was born out of the inability to simulate large ad hoc networks with existing tools, and some impatience in dealing with their internal complexity, which tended to make implementing new functionality a lengthy and bug-ridden task. NAB's design is node-oriented (and object-oriented); that is each node is represented by an object. It is written in OCaml and is actually the only simulator written in a language whose syntax is not derived from C. It is open source.

ns-2 [3] is the *de facto* standard for network simulation. Its behavior is highly trusted within the networking community. Its is developed at ISI, California, and is supported by the DARPA and NSF. *ns-2* is a discrete-event simulator organized according to the OSI model and primarily designed to simulate wired networks.

The support for wireless networking had been brought by several extensions. The Monarch CMU projects [1] made available an implementation of the IEEE 802.11 layers (WiFi). The BlueHoc [45] and BlueWare [9] projects provided the Bluetooth layers.

ns-2 provides a set a randomized mobility models, including random waypoint. Advanced node mobility had been make available by the Graph Mobility project [68], the GEMM project [56], and the Obstacle Mobility [39][38] model. These constitute a progress towards realistic simulation.

The core of *ns-2* is a monolithic piece of C++ code. It is extendable by adding C++ modules. The configuration relies on OTCL (a dialect of TCL by MIT) scripts. *ns-2* then appears to the user as an OTCL interpreter. More precisely, it reads scenarios files written in OTCL and produces a trace file in its own format. This trace needs to be processed by user scripts or converted and rendered using the NAM tool.

Thanks to its open source licence and its popularity, new extensions are sporadically proposed. For example, Dricot and De Doncker [27] proposed a highly accurate physical model based on ray tracing and Markov chains. This extension, which can be very useful for MANETs simulation, makes the simulator to be about 100 times slower.

ns-2 is a sound solution to MANET simulation. Unfortunately it suffers for its lack of modularity and its inherent complexity (*ns-2* was candidate to be the basis for the Qualnet [51] simulator but got finally rejected). Indeed, adding components/protocols or modifying existing ones is not as straightforward as it should be. For a long time, *ns-2* has been said to have few good documentation. The situation recently changed, as several users have put online their experience in the form of tutorials or example-driven documentations.

Another well-known weakness of *ns-2* is its high consumption of com-

putational resources. A harmful consequence is that *ns-2* lacks scalability, which impedes the simulation of large networks (*ns-2* is typically used for simulations consisting of no more than a few hundreds nodes).

Several projects have aimed at improving *ns-2*'s runtime. For example, staged simulation [70] (see section 3.2.2) and parallelism (see section 3.2.1 and next item) have turned out to be efficient solutions.

pdns [60] is developed at Gorgia Tech institute, California. The Parallel/Distributed Network Simulator aims at overcoming the limitation of *ns-2* regarding its scalability. *pdns* boosts *ns-2* processes by distributed the simulation over a network of closely coupled workstations (a common TCP/IP is usable). More precisely, it achieves an efficient parallelization of the simulation process by making distinct instances of *ns-2* simulating distinct sub-networks. *pdns* can simulate networks consisting of up to hundreds of thousands nodes.

OMNet++ [36] is a well-designed simulation package written in C++. OMNET++ is actually a general-purpose simulator capable of simulating any system composed of devices interacting with each others. It can then perfectly be used for MANETs simulation. The mobility extension for OMNeT++ [28] is intended to support wireless and mobile simulations within OMNeT++. This support is said to be fairly incomplete. OMNet++ is for academic and educational use.

OPNet [26] (Optimized Network Engineering Tools) is a discrete-event network simulator first proposed by MIT in 1986. It is a well-established and professional commercial suite for network simulation. It is actually the most widely used commercial simulation environment. OPNET Modeler features an interactive development environment allowing the design and study of networks, devices, protocols, and applications. For this, an extensive list of protocols are supported. Particularly, MAC protocols include IEEE 802.11a/b/g and Bluetooth ones. One of the most interesting features of OPNet is its ability to execute and monitor several scenarios in a concurrent manner. In spite of its wide adoption, some doubts remain regarding the dependability of its MANETs simulation engine. More precisely, Cavin and al. [21] simulated a broadcasting process on the OPNet, *ns-2* and GloMoSim simulators (each of them is detailed in this section). It came that the results obtained using OPNet were barely comparable to those harvested out of *ns-2* and GloMoSim, which exhibited similar behaviors. The divergences were quantitative but also qualitative (not the same general behavior). OPNet is written in C++.

QualNet [51] is a commercial ad hoc network simulator based on the GloMoSim core. It extends the GloMoSim offer by bringing support, a decent documentation, a complete set of user-friendly tools for building scenarios and analyzing simulation output. QualNet also largely extends the set of models and protocols supported by the initial GloMoSim distribution.

As it is built on top of GloMoSim, QualNet is written in Parsec [8].

SWANS [10], developed at Cornell university, is a Java-based wireless network simulator built atop the JiST discrete event platform. SWANS boasts a highly efficient sequential simulation engine and has been compared to GloMoSiM, in terms of quality. JiST relies on the concept of virtual-machine simulation. The way SWANS implements simulated time is singular: the simulated time is not managed by some shared clock. Instead, each entity (referred to as a *TimeFull* entity) is in charge of determining the time needed to its execution. By invoking each TimeFull entities in a sequence, SWANS gets able to determine the current simulated time. SWANS appears to the user as a framework. It must be programmed in plain Java, using some specific programming interface. SWANS is developed in Java and is open source.

5 Which simulator for what need?

MANETs simulators exhibit different features and models. The choice of a simulator should be driven by the requirements.

Determining the level of details required is key. If high-precision PHY layers are needed, then *ns-2* (coupled with the highly-accurate PHY [27]) is clearly the wisest choice. On the contrary, if the wireless technology has not impact on the targeted protocol, recent simulators (like NAB or Jane) which propose high-level abstractions and polished object-oriented designs will be more adapted.

The number of nodes targeted also determines the choice of the simulation tool. Sequential simulators should not be expected to run more that 1,000 nodes. If larger scales are needed, then parallel simulators are a wise choice. You may also consider highly optimized simulators like *ns-2* coupled with stage simulation.

Finally, most non-commercial simulators suffer from a lack of good documentation and support. Using a commercial one might help in case of troubles. Moreover, commercial simulators usually feature extensive lists of supported protocols, while open source solutions give full empowerment.

6 Perspectives and conclusion

Since the availability of the IEEE 802.11 (WiFi) standard, researchers investigate mobile ad hoc networks (MANETs). Since then, many simulators were proposed, less than twenty are still active projects. Some are dedicated to MANETs simulation [37][29][10] and some others consist in extensions of wired network simulators [3][73] and general-purpose discrete-event simulation engines [8][7]. As the needs of researchers continue to evolve, it is likely that existing simulators will integrate new functionalities and concepts as well as fresh simulators will be developed.

By looking at the current status of MANETs simulation and networking technologies, we envision that the follwing trends will drive the future developments.
First, the success of beowulf clusters and grid-computing has an impact on parallel/distributed simulators. Initially designed to run on SMP computers, simulators should now benefit from the huge distributed computational power offered by grid-computing facilities. The recent progress done in the field of multi-agent platform (MAPs) should also have a significant impact on distributed simulation. More precisely, new efficient algorithms for load-balancing and in distributed systems [12] turn the MAPs to appealing distribution frameworks.
Second, too few simulators facilitate the migration of the simulation code to real devices [29][46]. The code then needs to be written twice. Either common APIs for "ad hoc programming" should be strictly followed or simulators should also provide execution environments allowing the simulation code to be directly executable on devices.
Third, it is very likely that MANETs will be deployed within the metropolitan environment [23]. Project proposing constrained mobility models [56][67][38] were a first step towards realistic models. New generation simulators [37][29] natively integrate some of the properties of metropolitan mobility. It is very likely that coming simulators/extensions will integrate recent studies on radio propagation in the metropolitan environment [16] and will hence constitute the first generation of "metropolitan MANETs simulators".

References

[1] "The CMU Monarch Project's Wireless and Mobility Extensions to NS. http://www.monarch.cs.cmu.edu/," .

[2] "The COMPASS project. http://www.cc.gatech.edu/computing/compass/," .

[3] "The network simulator. *ns-2*. http://www.isi.edu/nsnam/ns," .

[4] America, P. and F. van der Linden, *A parallel object-oriented language with inheritance and subtyping*, in: *OOPSLA/ECOOP*, 1990, pp. 161–168.

[5] Arlitt, M., Y. Chen, R. J. Gurski and C. L. Williamson, *Traffic modeling in the ATM-TN telesim project: Design, implementation, and performance evaluation*, Technical Report DR-95-6 (1995).

[6] Baezner, D. and al, *Sim++: The transition to distributed simulation*, in: *Proceedings of the SCS Multiconference on Distributed Simulation*, 1990, pp. 211–218.

[7] Bagrodia, R. and W.-T. Liao, *Maisie: A language for the design of efficient discrete-event simulations*, IEEE Tranactions in Software Engineering **20** (1994), pp. 225–238.

[8] Bagrodia, R., R. Meyer, M. Takai, Y. an Chen, X. Zeng, J. Martin and H. Y. Song, *Parsec: A parallel simulation environment for complex systems*, Computer **31** (1998), pp. 77–85.

[9] Barr, R., *Blueware: Bluetooth simulator for ns*, Technical report, MIT, Cambridge (2002).

[10] Barr, R., *An efficient, unifying approach to simulation using virtual machines*, in: *PhD thesis*, 2004.

[11] Barr, R., Z. J. Haas and R. van Renesse, *Jist: Embedding simulation time into a virtual machine*, in: *EuroSim Congress on Modelling and Simulation*, 2004.

[12] Bertelle, C., A. Dutot, F. Guinand and D. Olivier, *Dynamic placement using ants for object based simulations*, in: *CoopIS/DOA/ODBASE*, 2003, pp. 1263–1274.

[13] Bettstetter, C., *Smooth is better than sharp: a random mobility model for simulation of wireless networks*, in: *MSWIM '01: Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems* (2001), pp. 19–27.

[14] Bettstetter, C., G. Resta and P. Santi, *The node distribution of the random waypoint mobility model for wireless ad hoc networks*, IEEE Transactions on Mobile Computing **2** (2003), pp. 257–269.

[15] Bohacek, S. and V. Sridhara, *The graph properties of manets in urban environments*, in: *(In Submission)*, 2004.

[16] Bohacek, S. and V. Sridhara, *The udel models - manet mobility and path loss in an urban*, in: *(In Submission)*, 2004.

[17] Booth, C. J. M. and D. I. Bruce, *Stack-free process-oriented simulation*, in: *PADS '97: Proceedings of the eleventh workshop on Parallel and distributed simulation* (1997), pp. 182–185.

[18] Buszko, D., W.-H. D. Lee and A. S. Helal, *Decentralized ad-hoc groupware api and framework for mobile collaboration*, in: *GROUP'01: Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work* (2001), pp. 5–14.

[19] Camp, T., J. Boleng and V. Davies, *A Survey of Mobility Models for Ad Hoc Network Research*, Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications **2** (2002), pp. 483–502.

[20] Catedra, M. F. and J. Perez, "Cell Planning for Wireless Communications," Artech House, Inc, 1999.

[21] Cavin, D., Y. Sasson and A. Schiper, *On the accuracy of manet simulators*, in: *POMC '02: Proceedings of the second ACM international workshop on Principles of mobile computing* (2002), pp. 38–43.

[22] Computing Laboratory, N. U., "JavaSim's Users Guide. http://javasim.ncl.ac.uk," .

[23] Conti, M., S. Giordano, G. Maselli and G. Turi, *Mobileman: Mobile metropolitan ad hoc networks*, in: *Proceedings of the 8th International IFIP-TC6 Conference, Lecture Notes in Computer Science LNCS 2775*, 2003, pp. 194–205.

[24] Couto, D. S. J. D., D. Aguayo, B. A. Chambers and R. Morris, *Performance of multihop wireless networks: shortest path is not enough*, SIGCOMM Computer Communications Rev. **33** (2003), pp. 83–88.

[25] De, P., A. R. S. Sharmaand and T. cker Chiueh, *Design considerations for a multi-hop wireless network testbed*, In Submission .

[26] Desbrandes, F., S. Bertolotti and L. Dunand, *Opnet 2.4: An environment for communication network modeling and simulation*, in: *Proceedings of European Simulation Symposium. Society for Computer Simulation*, 1993, pp. 64–74.

[27] Dricot, J.-M. and P. D. Doncker, *High-accuracy physical layer model for wireless network simulations in ns-2*, in: *IWWAN'04: Proceedings of the International Workshop on Wireless Ad-hoc Networks*, Oulu, Finland, 2004.
URL http://cs.ulb.ac.be/publications/P-04-05.pdf

[28] Drytkiewicz, W., S. Sroka, V. Handziski, A. Koepke and H. Karl, *A mobility framework for omnet++*, in: *3rd International OMNeT++ Workshop, at Budapest University of Technology and Economics, Department of Telecommunications Budapest, Hungary*, 2003.

[29] Frey, H., D. Görgen, J. K. Lehnert and P. Sturm, *A java-based uniform workbench for simulating and executing distributed mobile applications*, Scientific Engineering of Distributed Java Applications (2003).

[30] Frey, H., J. K. Lehnert and P. Sturm, *Ubibay: An auction system for mobile multihop ad-hoc networks*, in: *Workshop on Ad hoc Communications and Collaboration in Ubiquitous Computing Environments*, 2002.

[31] Görgen, D., H. Frey and C. Hutter, *Information dissemination based on the en-passant communication pattern*, KiVS: Fachtagung Kommunikation in Verteilten Systemen (2005).

[32] Heidemann, J., N. Bulusu, J. Elson, C. Intanagonwiwat, K. Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan, *Effects of detail in wireless network simulation*, in: *Proceedings of the SCS Multiconference on Distributed Simulation*, 2001, pp. 3–11.

[33] Hellbrück, H. and S. Fischer, *Towards analysis and simulation of ad-hoc networks*, in: *ICWN02: Proceedings of the International Conference on Wireless Networks*, Las Vegas, Nevada, USA, 2002, pp. 69–75.

[34] Hogie, L., P. Bouvry and F. Guinand, "The Madhoc simulator http://www-lih.univ-lehavre.fr/~hogie/madhoc/," .

[35] Hong, X., M. Gerla, G. Pei and C.-C. Chiang, *A group mobility model for ad hoc wireless networks*, in: *MSWiM '99: Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems* (1999), pp. 53–60.

[36] Imre, S., Keszei, Horváth, Hollós, Barta and Kujbus, *Simulation environment for ad-hoc networks in omnet++*, in: *IST Mobile Summit 2001*, 2001, pp. 135–140.

[37] Information Sciences Institute, E., *The nab (network in a box) wireless network simulator*, in: *http://nab.epfl.ch*, 2004.

[38] Jardosh, A., E. M. Belding-Royer, K. C. Almeroth, and S. Suri, *Real world environment models for mobile ad hoc networks*, Journal on Special Areas in Communications - Special Issue on Wireless Ad hoc Networks **14** (2005).

[39] Jardosh, A., E. M. Belding-Royer, K. C. Almeroth and S. Suri, *Towards realistic mobility models for mobile ad hoc networks*, in: *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking* (2003), pp. 217–229.

[40] Jetcheva, J., Y.-C. Hu, S. PalChaudhuri, A. K. Saha and D. B. Johnson, *Design and evaluation of a metropolitan area multitier wireless ad hoc network architecture*, in: *WMCSA*, 2003, pp. 32–43.

[41] Johansson, P., T. Larsson, N. Hedman, B. Mielczarek and M. Degermark, *Scenario-based performance analysis of routing protocols for mobile ad-hoc networks*, in: *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking* (1999), pp. 195–206.

[42] Kaba, J. T. and D. R. Raichle, *Testbed on a desktop: strategies and techniques to support multi-hop manet routing protocol development*, in: *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing* (2001), pp. 164–172.

[43] Klein, M., *Dianemu: A java based generic simulation environment for distributed protocols*, in: *Technical Report. Universitat Karlsruhe, Faculty of Informatic*, 2003.

[44] Kunze, C., U. Grossmann, W. Storka and K. Muller-Glaser, *Application of ubiquitous computing in personal health monitoring systems*, in: *DGBMT: Jahrestagung der Deutschen Gesellschaft Für Biomedizinische Technik*, 2002, pp. 360–362.

[45] Lab, I. I. R., "BlueHoc," .
URL http://www-124.ibm.com/developerworks/opensource/bluehoc

[46] Lehnert, J. K., D. Görgen, H. Frey and P. Sturm, *A scalable workbench for implementing and evaluating distributed applications in mobile ad hoc networks*, in: *WMC'04: Western Simulation MultiConference*, 2004, pp. 154–161.

[47] Lu, S. and J. A. Schormans, *Simulation acceleration techniques for mobile ad hoc networks*, in: *London Communications Symposium* (2003).

[48] Lundgren, H., D. Lundberg, J. Nielsen, E. Nordstr"om and C. Tschudin, *A large-scale testbed for reproducible ad hoc protocol evaluations*, in: *WCNC: 3rd annual IEEE Wireless Communications and Networking Conference* (2002), pp. 412–418.
URL
http://www.it.uu.se/research/core/publications/lundwcnc2002.pdf

[49] McNab, R. and F. Howell, *Using java for discrete event simulation*, in: *UKPEW: 20th UK Computer and Telecommunication Performance Engineering Workshop*, pp. 219–228.

[50] Naoumov, V. and T. Gross, *Simulation of large ad hoc networks*, in: *MSWIM '03: Proceedings of the 6th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems* (2003), pp. 50–57.

[51] Networks, S., "Qualnet user manual. http://www.scalable-networks.com/products/qualnet.php," .

[52] Pei, G., M. Gerla, X. Hong and C.-C. Chiang, *A wireless hierarchical routing protocol with group mobility*, in: *WCNC1999; IEEE Wireless Communications and Networking Conference*, 1, IEEE (1999), pp. 1538–1542.
URL http://www.cs.ucla.edu/NRL/wireless/PAPER/wcnc99.pdf.gz

[53] Perumalla, K., R. Fujimoto and A. Ogielski, *Ted — a language for modeling telecommunication networks*, SIGMETRICS Perform. Eval. Rev. **25** (1998), pp. 4–11.

[54] Polley, J., D. Blazakis, J. McGee, D. Rusk and J. S. Baras, *Atemu: A fine-grained sensor network simulator*, in: *SECON '04: Proceedings of First IEEE International Conference on Sensor and Ad Hoc Communication Networks*, Santa Clara, CA, 2004.

[55] Preiss, B. R., *The Yaddes distributed discrete event simulation specification language and execution environments*, Proceedings of the SCS Multiconference on Distributed Simulation **21** (1989), pp. 139–144.

[56] Ray and Suprio, *Realistic mobility for manet simulation* (2004).

[57] Riera, S. M., O. Wellnitz and L. Wolf, *A zone-based gaming architecture for ad-hoc networks*, in: *NETGAMES '03: Proceedings of the 2nd workshop on Network and system support for games* (2003), pp. 72–76.

[58] Riley, G. and M. Ammar, *Simulating large networks: How big is big enough?*, in: *Proceedings of First International Conference on Grand Challenges for Modeling and Simulation*, 2002.

[59] Riley, G. F., *The georgia tech network simulator*, in: *MoMeTools '03: Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research* (2003), pp. 5–12.

[60] Riley, G. F., R. M. Fujimoto and M. H. Ammar, *A generic framework for parallelization of network simulations*, in: *MASCOTS '99: Proceedings of the 7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (1999), p. 128.

[61] Ritter, H., M. Tian, T. Voigt and J. H. Schiller, *A highly flexible testbed for studies of ad-hoc network behaviour*, in: *LCN*, 2003, pp. 746–752.

[62] S. Park, A. S. and M. B. Srivastava, *Sensorsim: a simulation framework for sensor networks*, in: *Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, Boston, MA USA, 2000, pp. 104–111.

[63] Sanghani, S., T. X. Brown, S. Bhandare and S. Doshi, *Ewant: The emulated wireless ad hoc network testbed*, in: *WCNC: IEEE Wireless Communications and Networking Conference*, 2003.

[64] Scheidegger, M., F. Baumgartner and T. Braun, *Simulating large-scale networks with analytical models*, in: *International Journal of Simulation Systems, Science and Technology Special Issue on: Advances In Analytical And Stochastic Modelling*, 2005.

[65] Schindelhauer, C., T. Lukovszki, S. Rührup and K. Volbert, *Worst case mobility in ad hoc networks*, in: *SPAA '03: Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures* (2003), pp. 230–239.

[66] Schwetman, H., *Csim18 — the simulation engine*, in: *WSC '96: Proceedings of the 28th conference on Winter simulation* (1996), pp. 517–521.

[67] Tian, J., J. Hähner, C. Becker, I. Stepanov and K. Rothermel, *Graph-based mobility model for mobile ad hoc network simulation*, in: *Annual Simulation Symposium*, 2002, pp. 337–344.

[68] Tian, J., J. Hähner, C. Becker, I. Stepanov and K. Rothermel, *Graph-based mobility model for mobile ad hoc network simulation*, in: *Annual Simulation Symposium*, 2002, pp. 337–344.

[69] Waldorf, J. and R. Bagrodia, *Moose: A concurrent object-oriented language for simulation*, Int. Journal in Computer Simulation **4** (1994), pp. 235–257.

[70] Walsh, K. and E. G. Sirer, *Staged simulation: A general technique for improving simulation scale and performance*, ACM Trans. Model. Comput. Simul. **14** (2004), pp. 170–195.

[71] Williams, B. and T. Camp, *Comparison of broadcasting techniques for mobile ad hoc networks*, in: *MOBIHOC: Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2002, pp. 194–205.

[72] Yoon, J., M. Liu and B. Noble, *Random waypoint considered harmful*, in: *INFOCOM: Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, 2003, pp. 1312–1321.

[73] Zeng, X., R. Bagrodia and M. Gerla, *Glomosim: A library for parallel simulation of large-scale wireless networks*, in: *Workshop on Parallel and Distributed Simulation*, 1998, pp. 154–161.

[74] Zhang, X. and G. F. Riley, *Bluetooth simulations for wireless sensor networks using gtnets*, in: *MASCOTS*, 2004, pp. 375–382.

[75] Zhang, Y. and W. Li, *An integrated environment for testing mobile ad-hoc networks*, in: *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing* (2002), pp. 104–111.