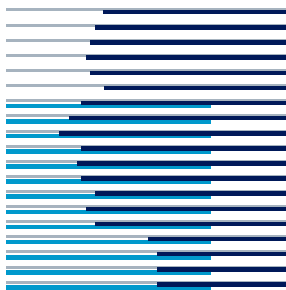


# Analysis of simulation environments for mobile ad hoc networks

Gianni A. Di Caro



**Technical Report No. IDSIA-24-03**

December 2003

**IDSIA / USI-SUPSI**

Dalle Molle Institute for Artificial Intelligence

Galleria 2, 6928 Manno, Switzerland

---

This report has been directly derived from Deliverable D11 "*Demonstrator for mobile ad hoc networks*" of the EU-funded RTD project BISON (IST-2001-38923)

IDSIA is a joint institute of both University of Lugano (USI) and University of Applied Sciences of Southern Switzerland (SUPSI), and was founded in 1988 by the "Dalle Molle" Foundation which promoted quality of life.



## **Abstract**

This document is directly derived from Deliverable D11 of the EU-funded project BISON [4]. It discusses the characteristics of simulation environments for telecommunications networks, and, more specifically, for mobile ad hoc networks. Different available network simulators are discussed and compared according to a number of criteria ranging from the reliability of the provided simulation models, to the degree of usability of software and graphical interfaces. The aim of the document is to identify the simulation environment which is the most appropriate to run experiments to test and validate the design of novel adaptive routing algorithms developed in the framework of BISON. However, most of the discussions and results hold in general. The document concludes that the (commercial) QualNet [21] simulator appears to be the most satisfactory product currently available.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>General characteristics and requirements of simulation studies</b>	<b>4</b>
<b>3</b>	<b>Simulation of mobile ad hoc networks</b>	<b>6</b>
3.1	Components of a mobile ad hoc network in the real-world . . . . .	6
3.2	Simulation model and selection of the level of detail . . . . .	7
<b>4</b>	<b>BISON's general requirements for the simulation environment</b>	<b>8</b>
<b>5</b>	<b>Selection criteria for network simulators</b>	<b>9</b>
<b>6</b>	<b>State-of-the-art of network simulators for mobile ad hoc networks</b>	<b>12</b>
6.1	OPNET . . . . .	12
6.2	GloMoSim/QualNet . . . . .	15
6.3	NS-2 . . . . .	18
6.4	OMNeT++ . . . . .	22
<b>7</b>	<b>Conclusion: The choice of QualNet as simulation framework</b>	<b>24</b>

# 1 Introduction

This document is directly derived from Deliverable D11 of the EU-funded project *BISON* [4]. It discusses the characteristics of simulation environments for telecommunications networks, and, more specifically, for mobile ad hoc networks. The aim of this document is to identify the simulation environment which is the most appropriate to run experiments to test and validate the design of novel adaptive routing algorithms developed in the framework of *BISON*. In spite of the focus on *BISON*'s specific objectives, most of the reported discussions and results hold in general.

*Discrete-event simulation* is the main tool used to study the characteristics and predict in some extent the behavior of communication networks, and, more in general to study complex stochastic dynamic systems modeling real-world situations of practical interest. Generally speaking, simulation is an extremely powerful and flexible tool. It potentially allows to study a large number of possible system configurations controlling the amount of complexity and realism included in the simulation model. The purpose of a simulation study consists to draw conclusions that are at the same time statistically sound, meaningful, and of practical interest. The exact definition of what aspects of the system under study should be included in the simulation model and their level of detail are the central design choices constraining the quality of the final output and of the derived conclusions.

Simulation is not the only tool available for the study of dynamic network systems, which are *BISON*'s main object of study. However, from a recent survey [19] it results that far more the 50% of the research results published during the last years in the major journals/proceedings in the field of telecommunications are obtained through the use of simulation. This situation can be explained both by the continually increasing computing power made available to the researchers to run extensive simulations and by the fact that *theoretical analysis* and *direct experimentation* are unfortunately of limited application. In fact, from one side, direct experimentation is costly and difficult to realize in a realistic way and without affecting the normal activities of the users. On the other side, meaningful theoretical analysis can be mainly carried out either for limit cases or after making working assumption which are seldom met in real-world conditions.

Nevertheless, considered the intrinsic complexity of telecommunication systems, made of a number of tightly interacting components of different nature (e.g., users, transmission links, processing/routing nodes, communication protocols), none of the available tools (simulation, theoretical analysis, and direct experimentation) alone is in principle sufficient in order to draw conclusions which are general, sound, and of practical interest. On the contrary, results coming from the application of the different tools should be integrated to get a more reliable picture of the ensemble of the characteristics associated to the network system under consideration. However, this document only focuses on simulation. More specifically, it considers the use of simulation and compares different simulators (QualNet, NS-2, OPNET, etc.) for the specific case of mobile ad hoc networks (MANETs). The ultimate purpose of this document is to identify the simulation environment which appears to be the most appropriate to study performance and behavior of novel routing algorithms for MANETs.

The documents is organized as follows. First, a general introduction to simulation is presented in Section 2. In Section 3 the characteristics of simulation models for MANETs are discussed.

Section 4 describes the specific BISON's requirements for a simulation environment. Section 5 discusses a number of criteria to score the different simulators and in Section 6 these criteria are applied to five different state-of-the-art simulators. The last section 7 discusses the reasons why QualNet has been preferred over the other simulators.

## 2 General characteristics and requirements of simulation studies

Generally speaking, *simulation* (e.g., [3, 2]) can be defined as the process of designing a model of a real system and conducting experiments with this model for the purpose of understanding the behavior of the system and/or evaluating various strategies for its control. To be this process useful, the behavior of the model is expected to faithfully mimic the response of the system under study.

Given the intrinsic discrete nature of the events happening in a communication network (e.g., start/end of user sessions), in BISON we are mainly interested in *discrete-event* simulations. In a discrete-event simulation model, state variables only change in correspondence of events happening at discrete points in time. Events occur as a consequence of activities and delays. Active simulation elements may compete for system resources, possibly waiting in queues to get access to an available resource. A discrete-event simulation model runs over time through a mechanism that moves simulated time forward according to the sequence of events. The system state is updated at each event along with the capturing and/or freeing of resources that may occur at that time.

A number of logical and practical steps have to be followed in any simulation study [3]. In particular, three major phases of the process of building and using a simulation model can be identified: (i) *definition of the model* of the real system, (ii) *collection of data* from the real system, and (iii) *definition of the experimental design* and of the actual *run* of the simulation using the defined model.

**Definition of the characteristics of the simulation model.** The first phase concerns the definition of the characteristics of the simulation model in relationship to: (a) the characteristics of the chosen abstract *representation* of the real system under investigation, (b) the objectives of the study. Aspect (a) plays the role of a *meta-model*. There is no a unique or even a "more natural" way of representing a whatever system. The experimenter implicitly or explicitly has to decide the resolution level which has to be considered for the system, possibly as a function of the ultimate objectives of the study. The meta-model defines which are the "atomic" parts of the system in terms of entities, interactions and structures. Informally speaking, the meta-model is how the reality is seen through the lenses of the experimenter.

The simulation model is in turn designed on the basis of the adopted meta-model. The simulation models that we are mainly considering here are *microscopic* (or *mechanistic*) ones, and not *macroscopic* (or *phenomenological*) ones. In general, the use of models at multiple scales, ranging from microscopic to macroscopic, can serve to investigate different properties of the same system. It is however important to be able to understand which is the right level of detail *necessary* for the purpose of the study at hand. The inclusions of unnecessary details can at the same time hide interesting aspects and waste computing

resources. On the contrary, when adopting a too low level of detail, the derived conclusions can easily bear no practical relationship with reality.

**Collection of input/output data from the real system.** The second phase concerns with the practical acquisition of *input* and *output data* from the real system. Input data are necessary to set the main parameters of the model (e.g., inter-arrival rates of traffic sessions, processing times, failure rates), while output data are necessary to *validate* the significance of the obtained results.

**Definition of the experimental design and run of the simulations.** The third, final phase, addresses the issue of the definition of the whole experimental design and of the other necessary operational details: the number of runs, the duration of the simulations, the measures of effectiveness to be used, the data that have to be collected, the characteristics of the random number generators, etc. The experiments must be designed such that the observed results can provide *statistically sound* answers to the stated objectives of the study. Accordingly, available computing resources must be adequate to run the selected number of simulations under the defined experimental conditions.

In general, it is not a trivial task to design and run a simulation model which can provide results significant and statistically sound for the system under consideration. This is especially true for the case of telecommunication networks, given the intrinsic complexity and variety of telecommunication systems, as well as the major role played in these systems by the users, whose behavior and expectations are hard to predict and model. In some sense, it is reasonable to expect that slightly different conceptual models of the same physical network can easily generate dramatically different simulation results. In general, this is an intrinsic characteristic of highly non-linear complex systems. Such a situation asks for simulations which are carefully designed in all their aspects, and in particular concerning with their statistical soundness, comparison issues, and overall validation. Unfortunately, in [19] it is claimed that this does not happen so often as it would be required, and the issue of the general *credibility* of the final results of simulation studies is pointed out. On the other hand, in spite of the many, intrinsic, difficulties, simulation is still the primary tool for the study of telecommunication networks and in BISON we are going to heavily rely on it. It is anyway important to be aware of the *limitations* and of the problems related to the simulation approach in order to avoid the most common methodological errors, when possible. This is the attitude we want to have in BISON.

In the case of mobile ad hoc networks, the limitations and the problems related to the use of simulation are even amplified with respect to the case of other types of networks (e.g., wired networks). In the following sections we discuss the characteristics of a generic mobile ad hoc network, pointing out the fact that there is really a wide spectrum of possible choices to build microscopic models for these networks. In fact, from one side, mobile ad hoc networks are made of a number of different components such that they are inherently complex. From the other side, in practice at the current moment there are no real-world implementations of mobile ad hoc networks, that could play the role of *reference* models and limit the range of the possible alternatives. This also means that the steps in the simulation process concerning with the acquisition of input/output data from the real system and the overall validation of the model cannot be carried out at all, or can only be done partially. Clearly, this situation is a bit embarrassing from a methodological point of view.

In spite of these difficulties, some simple reference models have *emerged* in the current literature. Most of the research works are based on the use of just two simulators (NS-2 and GloMoSim) offering a limited set of pre-built models, and on the choice of focusing on a set of simple, not really realistic, scenarios. In fact, it is common practice to consider networks with only few tenth of nodes moving quite slowly (or excessively fast) according to uncorrelated decision schemes in small flat open space areas. In BISON we plan to start using these same models to easily compare the performance of our algorithms with other approaches, but at the same time we will also move toward more realistic application scenarios and simulation models. At this aim, we need a simulation environment which is flexible enough to run tests over a wide range of different scenarios in terms of structural, dynamic, and physical characteristics.

### 3 Simulation of mobile ad hoc networks

#### 3.1 Components of a mobile ad hoc network in the real-world

In order to study the characteristics of a function (basic or advanced, in the BISON terminology) in a mobile ad hoc network, it is customary first to define what we mean by an instance of a mobile ad hoc network. We need to define our abstract representation (the meta-model, using the previous terminology) of such class of networks, making explicit all the components that can be considered in the simulation model. Our representation includes the following main components.

- A set of mobile devices, called *nodes*, with processing and wireless transmission capabilities powered by on-board batteries. The number of nodes participating to the network is in general not constant, at any time nodes can leave or join the network. In addition to the transmission capabilities, nodes can be also equipped with devices for geographic localization (e.g., GPS) or other optional devices of different nature.
- Since nodes are communication devices, their internal architecture is defined in terms of the *OSI protocol stack*. This means that node communications are regulated by the characteristics associated to the OSI protocol layers: physical, data link, medium access control (MAC), network, transport, session, presentation, application.
- Communications happen through a *wireless* interface. Therefore, the the physical and technological aspects of *radio emission, propagation and reception* has to be explicitly taken into account. While emission and reception are strictly related to the physical characteristics of the node radio device, radio propagation is affected by the physics of the environment in which the network is placed.
- The network is seen as embedded in an *environment* with well defined physical characteristics which affect both the propagation of the radio signals and the mobility patterns of the nodes.
- Every node is associated to a single user which moves inside the environment. Therefore, node *mobility patterns* depend on the characteristics of both the users and the environment.

- Node *energy power* relies on the use of on-board batteries. Every action involving the use of the radio channel, as well as the same fact that the device is switched on, has an energetic cost which affects the available energy power.
- Users generate *data traffic* in terms of open sessions between pairs or groups of users. Traffic patterns can in general assume arbitrary forms. The same node can open several traffic sessions during the length of the observation period. The generation of the traffic patterns is seen as a non-terminating process [8], in the sense that the network operations do not have “natural” starting and ending points.

### 3.2 Simulation model and selection of the level of detail

It is well understood that in general terms all the physical and logical components listed in our abstract representation are equally important to determine the overall dynamics of the network system. Therefore, all these components should be also included in our simulation model. However, the open question is: which is the right *level of detail* and which are the core specific characteristics that shall be used to model the different network components inside the simulation architecture? For instance, do we need to implement all the OSI layers and let every packet passing through them at each network node, or, is it sufficient to limit ourselves to the MAC, network, and session layers? How faithful has to be the implementation of each selected OSI layer? Is it a meaningful choice to model the environment simply as a small open flat area given that it is hard to think of a possible application scenario with these characteristics? Must the mobility of the nodes reflect some precise model of human behavior, or, can it be modeled in terms of little more than a random walk? Do the transmission range and the modality of access to the medium have to be adapted to the values of devices currently available on the market, or, can it be envisaged the use of smarter and more powerful technologies? Should the input traffic patterns be defined in the generic terms of negative exponential distributions, or, should they be close to the patterns usually encountered in GSM or other wireless networks? And so on ... The “appropriate” choice always depends on several factors starting from the specific objectives of the research.

In the case of wired networks, the extensive experience accumulated over the last 30 years allows nowadays to single out the relative importance of the different components and allows to obtain quite sound simulation results even when adopting a high level of abstraction. On the other hand, the younger field of mobile ad hoc networks provides less guidance on what abstractions are appropriate. Low-level details can have a large effect on performance, but detailed simulations (e.g., with faithful radio propagation models) can be very expensive. Again, the absence of practical real-world implementations complicates the whole issue. Finding a good trade-off between *accuracy*, *statistical soundness* (which needs the ability to run the simulations several times) and *practical/scientific interest* is not a trivial task.

Several arguments can be raised pro and against the use of detailed versus abstract simulations (good discussions on the subject can be found in [11, 9]). The use of detailed/realistic models in principle can truly help to understand the phenomena under study. In particular, since telecommunication networks are artificially built, it is possible to reverse engineer and bring inside the model the correct functioning behavior of the single hardware and software components. In some sense, this is a fortunate situation with respect to the modeling of natu-

ral systems. On the other hand, it is quite clear to practitioners in the field that a *fully realistic* microscopic simulation is not possible.

A simple example to stress the importance of using a high level of detail concerns the well-known *hidden terminal effect*, which cannot be evidenced without considering concurrent transmission and a quite realistic implementation of the access strategy to the transmission medium. On the other side, there are several reasons for intentionally choosing a high level of abstraction. Distillation of a research question to its essence can provide useful insights cleared up from specific details. When exploring a new area where many issues are unclear, as it is the case for mobile ad hoc networks, the need to quickly explore and compare a variety of alternatives can be more important than a detailed result for a single specific, realistically designed, scenario. A more abstract simulation can also make the effects of a change in the algorithm distinct, where they would be obscured by other effects in more detailed simulations. Moreover, as it is intuitive, omission of details can improve computational performance by multiple orders of magnitude (e.g., [13]). This is a critical argument, since better computational performance can allow simulations to *scale up*, as well as to consider more realistic (computationally-demanding) scenarios. Considering that the relative performance of ad hoc routing protocols seem to differ with the increasing of the number of nodes [10], the issue of scaling simulations appears even more fundamental. Clearly, on the other side, the lack of detail can also cause answers which are either simply *wrong* or *inapplicable* (technically correct but providing an answer to part of the design space that may not be sensible or relevant).

A number of studies [11, 9, 12, 14, 6] can be found in literature concerning the impact of different levels of details and of different models for the different components of a simulation. For instance, in [11] five different cases are studied, considering the impact of detail in relationship to the characteristics of the energy consumption, radio propagation, and data visualization models. In [6] the performance evaluation of ad hoc routing protocols are evaluated in function of different possible models of node mobility. The effect of different modeling design in the MAC layer, as well as, in the setting of the related parameters, has been extensively studied (e.g., [25, 5]). All these examples refer to the analysis of the impact of the modeling detail for one single aspect (energy, radio propagation, mobility, or MAC protocol). Other studies compare the answers provided by *different simulators*, that is, taking into account the whole set of aspects relevant for the study of mobile ad hoc networks. In [7], the performance of the same simple flooding algorithm is compared using three different simulators widely in use (NS-2, OPNET and GloMoSim).

## 4 BISON's general requirements for the simulation environment

All the mentioned studies and previous discussions confirm that little modeling differences can actually result in quite different behaviors, and, consequently, in possibly different conclusions. This does not imply that the use of simulation is meaningless. Simulation is still the most powerful tool to investigate the practical behavior of communication networks. Nevertheless, extreme care must be adopted selecting/implementing the characteristics of the simulator and at the moment of deriving conclusions. And it is extremely important to be *aware* of the potential problems and inconsistencies.

According to BISON's objectives and to the characteristics of the still young research field of

mobile ad hoc networks, it is clear that we have to look for a simulation architecture which is flexible enough to easily accommodate different, selectable, levels of detail. We intend to study the systems at different scales of detail, and we want to leave open the possibility to use, for a same level of detail, different sets of operational models for the single components of the mobile ad hoc network.

In this perspective, we have to look for an *open* and *modular* simulation architecture. We ruled out the choice of developing from *scratch* a new custom simulator. A satisfactory and well done job in this sense would take some time, which is not worth to spend since a number of good network simulators, both commercial and public domain, are already available from the Internet and from software telecommunication companies. We do *not* need “yet-another-simulator” from scratch. We can conveniently use an available simulator as the *initial platform* on which we can build the complete BISON simulation architecture for mobile ad hoc networks. It is in this sense that we need a truly open and modular system. BISON addresses specific issues and has specific requirements that we will bring inside the chosen simulation platform, expanding and modifying it.

Broadly speaking, discrete-event simulators for mobile ad hoc networks (and, more in general, for networks) can be divided in two classes. Those which are designed to simulate anything that can be mapped to active components that can access local resources and communicate by passing messages, and those which are specifically designed as network simulators. These latter usually include detailed and hard-coded concepts about nodes, agents, protocols, links, packet representation, network addresses, etc. Moreover, they usually come with a set of important components (e.g., routing or transport protocols) already implemented. If this is clearly an advantage, at the same time it is usually more difficult to modify hard-coded aspects to possibly adapt them to more specific needs. Again, a good trade-off must be found between flexibility, modifiability and availability of tools/components. In the following we discuss the characteristics of a set of four simulators which we have selected as possible candidates. These simulators are either the simulators most widely in use in the telecommunication community or new simulators which are attracting an increasing interest in the community because of their particularly good software design and performance.

Before discussing and comparing the different candidate simulators, in the next section we introduce a set of core features that we will use to score and rank the examined simulators.

## 5 Selection criteria for network simulators

### Types of supported simulations

We are mainly interested in online discrete-event simulations. However, when possible, it might be interesting to carry out *trace-driven* simulation, that is, simulations using as input a time-ordered record of events (also called a “trace”) from a real system.

More in general, since we have the intention to run studies at different scales of resolution, it could be very useful in practice to have a simulator offering the possibility to switch off most of the microscopic details (e.g., the OSI chain inside the nodes) in order to carry out statistical-mechanics-like studies involving a large number of elements and the use of important stochas-

tic components. In the following we refer to this form of simulation as *generic Monte Carlo simulation*.<sup>1</sup>

Another possibility that can be envisaged is the *injection of live traffic* into the simulator and/or the injection of traffic from the simulator into the live network. Actually, these characteristics refer more to *emulators* than simulators. Some experiments in this sense for mobile ad hoc networks have been carried out in the framework of the Monarch Project at Carnegie Mellon University [14].

### **Supported computational platforms**

According to some heterogeneity among the BISON's members in terms of used platforms, it would be helpful to have a simulator running on different platforms, in particular on both *Linux* and *Windows* systems.

Since we expect heavy computational loads associated to the simulations, *distributed, parallel* or *multi-threading* processing capabilities could be possibly exploited if available.

### **Support for the generation of topologies**

All simulators have some mechanism for the *generation of network topologies*. Among the several possibilities, there are the use of special *script* or *configuration languages*, manually edited *numerical files*, and *graphical interfaces*. Some simulators do not easily permit the creation of *hierarchical topologies*, admitting only flat topologies. In some fortunate cases a special tool for the automatic controlled generation of *random topologies* is provided with the simulator.

### **Support for the generation and management of traffic profiles**

In order to generate data traffic, it is in general necessary to use *generators of data* according to some specific target distribution (e.g., Poisson or derived from the observation of real traffic). Good simulators usually come either with a wide set of such traffic generators.

On the other side, it is also important to have tools to *profile the generated traffic* and to compute the *statistics* necessary to draw conclusions.

### **Monitoring support**

During the execution it can be extremely useful to *monitor* the network dynamics on a per-flow, per-node, or, more in general, on the basis of some aggregation criteria. Monitoring can be helped by the availability of *graphical interfaces*.

The results of monitoring can be also dumped on files to generate *traces* that can be used for further comparisons or for re-play the simulation for more careful studies of the generated dynamics.

---

<sup>1</sup> There is a clear abuse of the term Monte Carlo [24, 23] here. In fact, also in the case of the use of discrete-event or trace-driven simulation we expect to heavily use a stochastic component in the generation of traffic and mobility patterns, and of other various aspects of the simulation.

## Modules for the OSI protocols, mobility models, and radio propagation

A simulator coming with a rich set of already implemented and tested *modules* is preferable to a simulator coming with no implemented modules. However, according to the previous discussions on the extreme sensitivity of the obtained results to the model characteristics, it is clear that not only the number but also the characteristics (quality) of the implemented modules are equally important.

The OSI modules that are really interesting for us are primarily those relative to *routing* protocols. It is also very important to have a reliable implementation of the *MAC* layer. The *physical* and *link* layer, if considered in the perspective of the radio emission/reception are also important. At the *application* or *session* layer it is probably enough to have simple FTP-like or CBR modules. For what concerns the *transport* layer, only a *best-effort* (unreliable) UDP-like service is probably sufficient. A TCP-like service would be in principle very useful in the context of mobile ad hoc networks, but unfortunately there are a number of difficult problems related with this issue. Therefore, we will likely not use any form of reliable transport protocol.

Modules implementing *mobility models* are usually rather simple to implement. However, it can be helpful to have some mobility module coming with the simulator, because it is likely that such module is being used also by other researchers, making comparison studies easier. This argument is valid in general. Modules coming with a simulator are likely to be used by several research groups. Therefore, they “automatically” become basic models of reference.

*Radio wave propagation* models coming with simulators are usually straightforward implementation of basic standard results from electromagnetism studies. An accurate propagation model should take into account the morphology and the physics of the environment. This would be overwhelming expensive in computational terms and incredibly complicate.

## Flexibility, modifiability, extensibility and scaling issues

It has been already stressed that we want an architecture such that it is possible to select a desired level of detail, components can be easily switched on and off, current models can be substituted or modified, new models can be included. Shortly, the simulator must a truly *open modular* architecture.

In particular, scalability is an critical issue. In principle all simulators allow to add more nodes and to increase the computational burden associated to the traffic patterns and to the other dynamic aspects of the network (mobility, appearing/disappearing of nodes, broken paths, etc.). However, in practice some simulators scale much better than others, that is, some simulators have some practical limitations that must be evidenced, since we might consider to study networks with large number of nodes.

## General usability

The *software design* of the simulator, as well as the *programming tools* required to use the simulator greatly affect the *usability* of the simulator. It is preferable to have a simulator requiring the use of tools and concepts already in the knowledge background of the BISON members.

The availability of *graphical interfaces* can speed up the operations on the simulator and are therefore seen as a general positive aspect.

Last but not least, the overall quality of the available *documentation* and *technical support* are important factors to be able to quickly and effectively learn how to use and control the simulator.

### **Level of acceptance of the simulator by the network community**

According to the discussed sensitivity of the obtainable results to the specific simulator used, it is clear that choosing a simulator widely in use and accepted by the community allows to produce results which are easier to compare with those present in the literature, and, in some sense, which are more easily “accepted” by the scientific community.

### **Type of software license**

It is pretty obvious that *public domain* simulators are in principle more “convenient”. However, *commercial* simulators must also be considered and selected if they offer a strategic and clear set of advantages over the available public domain ones.

## **6 State-of-the-art of network simulators for mobile ad hoc networks**

We identified *OPNET*, *GloMoSim/QualNet*, *NS-2* and *OMNeT++* as the best possible candidates to run simulation studies in MANETs. These simulators are representative of the state-of-the-art in the field of simulators for mobile ad hoc networks. In the following of this section we discuss the main characteristics of each of these simulators both at a general level and from the point of view of the selection criteria reported in the previous Section 5 and summarized in Table 1.

### **6.1 OPNET**

OPNET (Optimized Network Engineering Tools) is a commercial tool from OPNET Technologies Inc. [17] for modeling and simulation of communications networks, devices, and protocols. It features graphical editors and animation. It has been developed for almost 15 years. It is widely held to be the state-of-the-art in network simulation. OPNET is used by large companies. It is available for *free* for North-American universities (although this free license does not include any technical support).

It can simulate all kinds of wired and several wireless networks. An IEEE 802.11 compliant MAC layer implementation is also provided. Although OPNET is rather intended for companies to diagnose or reorganize their network, it is possible to implement one’s own algorithm by reusing a lot of existing components. Most part of the deployment is made through a *hierarchical graphic user interface*. Network with several *hundreds of nodes* can be managed.

The software comprises several tools and is divided in several parts: *OPNET Modeler* and *OPNET Planner*, the *Model Library*, and the *Analysis tool*. Features included in this generic simulator are an event-driven scheduled simulation kernel, integrated analysis tools for interpreting

Table 1: Possible characteristics of simulation environments for mobile ad hoc networks and their common implementation. These characteristics are considered in Section 6 to score the different network simulators.

Characteristics of simulators	Possible alternatives and implementations
<i>Supported simulation types</i>	Discrete-event, trace-driven, Monte Carlo
<i>Computational platforms</i>	Linux, Windows, parallel and distributed systems
<i>Topologies</i>	Flat, hierarchical, random
<i>Definition of topologies</i>	Script languages, data files, graphical interfaces
<i>Data traffic generation</i>	Sampling from probabilistic distributions, real data
<i>Traffic profiling</i>	Online data collection and statistical analysis tools
<i>Monitoring support</i>	Graphical interfaces, trace generation
<i>Modules for the OSI layers</i>	Routing algorithms, MAC, physical and link layers
<i>Mobility models</i>	Random walk, random waypoint, gauss-markov
<i>Models for radio propagation</i>	Open space with ground reflection, shadowing effects
<i>Modifiability and extensibility</i>	Open and modular software design
<i>Scaling</i>	Efficient management of memory and CPU resources
<i>Ease of use</i>	Programming tools, graphic interfaces, documentation
<i>Scientific acceptance</i>	Number of publications using the simulator
<i>Type of software license</i>	Commercial, public domain

and synthesizing output data, graphical specification of models and a hierarchical object-based modeling.

### The internal architecture

The modeling methodology of OPNET is organized in a hierarchical structure. At the lowest level, which is the one really customizable, `Process models` are structured as a *finite state machines*. State and transitions can be graphically specified *graphically* using state-transition diagrams, whereas conditions that specify what happens within each state are programmed with a C-like language called *Proto-C*. Process models, and built-in modules in OPNET (source and destination modules, traffic generators, queues, wireless, ...) are then configured with

menus and organized into data flow diagrams that represent nodes by using the graphical `Node Editor`. Using the graphical `Network Editor`, nodes and links are selected to build up the topology of a communication network. The `Analysis Tool` provides a graphical environment to view and manipulate data collected during simulation runs. Results can be analyzed for any network element.

`OPNET Planner` is an application that allows administrators to evaluate the performance of communications networks and distributed systems, without programming or compiling. Planner analyzes behavior and performance by discrete-event simulations. Models are built using a graphical interface. The user only chooses pre-defined models (from the physical layer to the application) from the library and sets attributes. The user can define new models, but he/she has to contact the MIL3's modeling service.

The `Modeling libraries` are included with `OPNET Modeler` and `OPNET Planner` and contain protocols and environments (e.g., ATM, TCP, IP, Frame Relay, FDDI, Ethernet, IEEE 802.11, support for wireless), link models such as point-to-point and bus, queuing service disciplines such as First-in-First-Out (FIFO), Last-In-First-Out (LIFO), priority non-preemptive queuing, shortest first job, round-robin or preempt and resume.

### **Specific support for mobile ad hoc networks**

From the point of view of predefined components specific for mobile ad hoc networks, the *Wireless* (local, GSM and ad hoc) module of OPNET comes with most of all the necessary components in terms of mobility and radio propagation models, as well as in terms of full protocol stack, including MAC (802.11) and popular routing algorithms like Adhoc On-demand Distance Vector (AODV) and Dynamic Source Routing (DSR). Featuring an open-source Longley-Rice model, the *Terrain Modeling* module can replicate a number of known atmospheric or topological conditions and their combined impact on signal propagation. The *Wireless* module features also *parallel simulation* capabilities.

### **General summary of OPNET's features**

OPNET is a well-established and highly professional product. Support for mobile ad hoc networks is essential but of high-quality and sufficient to carry out serious studies. However, the results reported in [7] are in this sense a bit puzzling. In the paper a study of the behavior of a simple flooding algorithm is carried using OPNET, NS-2 and GloMoSim. While the results provided by NS-2 and GloMoSim are rather similar between them, the results provided by OPNET are significantly different. It is definitely unclear which is the "right" behavior. . .

The graphical interface simplifies most of the routine operations, while the development of new models always requires the definition of a finite state machine. If this fact can be seen as a difficulty at the beginning, it is also true that finite state machine are undoubtedly an expressive and effective tool for modeling. In OPNET, differently from NS-2 and GloMoSim, it results quite easy to describe an application that bypasses part of the protocol stack. This aspect can be quite important to speed up and/or to reduce the level of unnecessary detail during the simulations.

The performance seem to scale quite well, but there are not enough data in the current literature to make a more precise statement in this sense in the context of mobile ad hoc networks.

## 6.2 GloMoSim/QualNet

GloMoSim is a scalable simulation *library* designed at UCLA Computing Laboratory to support studies of *large-scale network* models, up to millions of nodes, using *parallel* and/or distributed execution on a diverse set of parallel computers (with both distributed and shared memory). It has been designed for wireless and wired networks systems, but it currently supports only protocols for purely *wireless* networks. GloMoSim is a library for the C-based parallel discrete-event simulation language *PARSEC* [18]. One of the important distinguishing features of *PARSEC* is its ability to execute a discrete-event simulation model using several different asynchronous parallel simulation protocols on a variety of parallel architectures. *PARSEC* is designed to cleanly separate the description of a simulation model from the underlying simulation protocol, sequential or parallel, used to execute it.

GloMoSim is build according to the *OSI layered* approach. A *common, standard, API* between every two neighboring models on protocol stacks is predefined to support their composition. These APIs specify parameter exchanges and services between neighboring layers. This allows the rapid integration of models developed at different layers by different people. Actual operational code can also be easily integrated into GloMoSim (e.g., this has already been done importing the BSD implementation of the TCP suite inside the library). If all protocol models obey the strict APIs defined at each layer, it is be feasible to simply swap protocol models at a certain layer (say evaluate the impact of using CSMA rather than 802.11 at the MAC protocol) without having to modify the models for the remaining layers in the stack. That is, the modular implementation enables consistent comparison of multiple protocols and of different levels of detail at a given layer.

To run GloMoSim, the *PARSEC* compiler is needed. To develop new protocols in GloMoSim, the user should have some familiarity with *PARSEC*, but it is not required an expert-level knowledge. For most protocols it is sufficient to write purely C code with the addition of few *PARSEC* functions.

A number of protocols have been developed at each layer, and models of these protocols or layers can be deployed at different levels of granularity and detail. are currently available in GloMoSim. Telnet, ftp, CBR, HTTP, replicated file system for what concerns applications. While at the transport layer the following models are available: TCP (FreeBSD), UDP, TCP (Tahoe). For unicast Routing the available algorithms are: AODV, Bellman-Ford, DSR, Fisheye, Flooding, LAR, NS DSDV, OSPF, WRP. The models available for the MAC layer are CSMA, IEEE 802.11, FAMA, MACA. The radio interface uses omni-directional antennas with and without capture capability. Radio wave propagation can be modeled according to Free space, Rayleigh, Ricean, or SIRCIM models (including obstructions and building-type situations). Mobility models include different variations of the basic Random waypoint, Random drunken, ECRV, BBN, Path-loss matrix, and group mobility.

A platform-independent tool coded in JAVA is available for both real-time and trace-based debug, monitoring and statistics report.

## The internal architecture

GloMoSim has a layered *architecture*. Simulation is a collection of network nodes, each with its own protocol stack parameters and statistics. In turn, each layer, is an object with its own variables and structures. Messages can be exchanged between *nodes* and *layers* at any desired level of grouping and granularity. The synchronization among the events is insured by self-scheduled (timer) messages. The library comes with a complete suite of simple functions, based on standard APIs, for the creation, transmission and manipulation of messages. New types of events, as well as of messages and node/layers can be created in a rather straightforward way. Adding a model or protocol to a layer requires: (i) an *initialization function* to allocate and initialize model specific data, (ii) a *finalization function* that defines the output statistics from the simulation run for the model, and (iii) a *simulation event handling function* that performs simulation actions when scheduled with an event. New user-defined function can be easily plugged into every layer by means of a skeleton interface code file to be modified by the user. In this way, users do not have to keep reinserting local changes in GloMoSim files for every GloMoSim version, and it makes easy to customize the interface file to call the new functions.

The internal architecture of GloMoSim has been designed with the specific aims to develop a *modular* simulation environment capable of *scaling* up to networks with thousands/millions of heterogeneous nodes and easy to port on a *parallel/distributed* computing environment. The requirements of scalability and modularity make the library design a challenging issue. GloMoSim assumes that the network is decomposed into a number of *partitions* and a *single* entity is defined to simulate a single layer of the complete protocol stack for all the network nodes that belong to the same partition. Interactions among the entities obey the corresponding APIs. Syntactically, the interactions may be specified using messages, function calls, or entity parameters as appropriate. This method supports modularity because a PARSEC library entity representing a layer of the protocol stack is largely self-contained. It encapsulates the complexity of a specific network behavior independently from other ones. This method also supports scalability because node aggregation inside one entity will be able to reduce the total number of entities, which has been found to improve also the sequential performance other than the parallel ones.

Efficient *parallel simulators* must address three general sets of concerns: efficient synchronization to reduce simulation overheads; model decomposition or partitioning to achieve load balance; efficient process to processor mappings to reduce communications and other overheads in parallel execution. GloMoSim offers several possibilities in this sense, but since at the moment we do not plan to use a parallel machine, we do not enter into details in this sense (see [27, 1] for results concerning the parallel performance of GloMoSim).

## QualNet: the commercial version of GloMoSim

GloMoSim is not public domain, but it is freely available without fee for education, or research, or to non-profit agencies. However, the documentation shipped with GloMoSim is quite poor (there is even no a user manual), as well as the set of tools available to speedup the generation of topologies, to monitor the system behavior, to analyze the post-simulation results, and, more in general, to design the characteristics of and interact with the whole simulation environment.

*QualNet* alleviates most of the GloMoSim's flaws. QualNet [21] is a commercial product from

Scalable Network Technologies which is derived from GloMoSim. However, QualNet has loads of additional features with respect to GloMoSim. QualNet comes with an extensive suite of faithful implementations of models and protocols for *both* wired and wireless networks (local, ad hoc, satellite and cellular), as well as extensive documentation and technical support. Three libraries are available: a Standard library which offers most of the models and protocols necessary for both research and business-oriented activities in the field of wired and wireless networks; a MANET library which provides additional and very specific components for ad hoc networks other than those already present in the standard library; and, a QoS library which includes specialized protocols for quality-of-service. QualNet also includes a *Digital Elevation Model (DEM)* to make nodes and radio waves moving in non-flat terrains with specified radio absorption characteristics. To our knowledge, QualNet seems to be the most complete network simulator, in terms of available protocols, models and tools for what concerns mobile ad hoc networks. It is hard to think of missing components. In spite of being a commercial product, most of the C source code of the included models and protocol is made available to customers favoring both a better understanding and a full customization.

The QualNet library is the basis for the QualNet Developer software suite which includes five different modules: Animator, which is a graphical tool for experiment setup (e.g., topology, layer protocols, traffic) and animation (e.g., watching of traffic flows and specific performance metrics, re-play of the simulation); the Designer, a finite state machine tool for custom protocol modeling; a state-based visual tool is used to define the events and processes of a new protocol model; Analyzer, a statistical graphing tool to report the statistics associated to hundreds of pre-defined or custom performance metrics; Tracer, which allows packet-level visualization for viewing the contents of a packet as it goes up and down the protocol stack; Simulator, the simulator itself, designed to accommodate high-fidelity models of networks of 10's of thousands of nodes with heavy traffic and high mobility. In addition, QualNet offers also Parallel Developer for managing and running parallel executions.

### **General summary of GloMoSim/QualNet's features**

GloMoSim has several nice features. It comes with a rich suite of models, it is the only simulator among the considered ones that seems able to scale up to thousands of nodes, its highly modular design is such that it appears quite straightforward to modify and/or extend the basic functionalities. The ability to scale up and the definition of standard APIs for the modular protocol stack, opens the possibility to flexibly investigate the effect of multiple models at dramatically different levels of detail.

Analysis and visualization tools are basic but sufficient for general studies. Documentation is quite poor.

GloMoSim is likely the second most used simulator, after NS-2, in the research community. Therefore, it is widely accepted from a scientific point of view.

QualNet is based on GloMoSim but it dramatically expands its capabilities in terms of contributed models and protocols, graphical tools for experiment planning, analysis and visualization, as well as, in terms of available documentation and technical support. QualNet supports also some form of network emulation and allows to design realistic 3D environments. We can say that QualNet is quite complete as a simulator.

### 6.3 NS-2

The NS network simulator [15], from U.C. Berkeley/LBNL, is a object-oriented discrete event simulator targeted at networking research and available as *public domain*. Its first version (NS-1) began in 1989 as a variant of the REAL network simulator [22] and was developed by the Network Research Group at the Lawrence Berkeley National Laboratory (LBNL), USA. Its development was then part of the VINT project [26], supported by DARPA, at LBNL, Xerox PARC, and UCB, under which NS version 2.0 (NS-2) was released, evolving substantially from the first version. The aim of the VINT was not to design a new network simulator, but to unify the effort of all people working in the research field of network simulation. The result is that NS-2 is widely used in the networking research community and has found large acceptance as a tool to experiment new ideas, protocols and distributed algorithms. Currently NS-2 development is still supported through DARPA. NS has always included substantial contributions from other researchers, including wireless code for both mobile ad hoc networks and wireless LANs from the UCB Daedalus and CMU Monarch projects and Sun Microsystems.

At the time being, NS-2 is well-suited for packets switched networks and wireless networks (ad hoc, local and satellite), and is used mostly for small scale simulations of queuing and routing algorithms, transport protocols, congestion control, and some multicast related work. It provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless networks.

NS-2 is suitable not only for simulation but also for *emulation*, that is, it is possible to introduce the simulator into a live network. Special objects within the simulator are capable of introducing live traffic into the simulator and injecting traffic from the simulator into the live network.

NS-2 play an important role in the research community of mobile ad hoc networks, being a sort of reference simulator. NS-2 is the most used simulator for studies on mobile ad hoc networks, and it comes with a rich suite of algorithms and models.

In this perspective, NS-2 would be the natural candidate to be used in BISON too. Unfortunately, its software architecture is such that adding new components and/or modifying existing ones is not a straightforward process. That is, in terms of *ease* to implement/test new algorithms or scenarios, NS-2 scores poorly with respect to other candidates. Moreover, NS-2 does *not scale* well in terms of number of nodes and it is reported to be in general quite slow from a computational point of view.

#### NS-2's internal architecture

The NS-2 architecture closely follows the *OSI model*. The *network model* represents the interconnection of network elements. It consists of nodes and links. Single or multiple traffic generators, including statistical generators and other typical generators such as FTP and Telnet, can be attached to any node. In addition, network and transport protocol behavior are simulated by attaching the appropriate agents to the interested nodes.

NS-2's code source is split between C++ for its core engine and OTcl [16], the MIT's object extension to Tcl, for configuration and simulation scripts. The combination of the two languages offers a sort of compromise between performance and ease of use. The package provides a compiled class hierarchy of objects written in C++ and an interpreted class hierarchy of ob-

jects written in OTcl related to the compiled ones. The user creates new objects through the OTcl interpreter. New objects are closely mirrored by a corresponding object in the compiled hierarchy. Tcl procedures are used to provide flexible and powerful control over the simulation (start and stop events, network failure, statistic gathering and network configuration). The OTcl interpreter provides commands to create the networks topology of links and nodes and the agents associated with nodes.

Implementation and simulation under NS-2 consists of 4 steps: (1) implementing the protocol by adding a combination of C++ and OTcl code to NS-2's source base; (2) describing the simulation in an OTcl script; (3) running the simulation and (4) analyzing the generated trace files. Implementing a new protocol requires adding C++ code for the protocol's functionality, as well as updating key NS-2 OTcl configuration files in order for NS2 to recognize the new protocol and its default parameters. The C++ code also describes which parameters and methods are to be made available for OTcl scripting.

The simulation is configured, controlled and operated through the use of interfaces provided by the OTcl class `Simulator`. The class provides procedures to create and manage the topology, to initialize the packet format and to choose the scheduler. It stores internally references to each element of the topology. The user creates the topology using OTcl through the use of the standalone classes `node` and `link` that provide a few simple primitives.

The function of a *node* is to receive a packet, to examine it and map it to the relevant outgoing interfaces. A node is composed of simpler `classifier` objects. Each `classifier` in a node performs a particular function, looking at a specific portion of the packet and forwarding it to the next `classifier`.

*Agents* are another important type of components of a node: they model endpoints of the network where packets are constructed, processed or consumed. Users create new sources or sinks from the class `Agent`. NS currently supports various TCP agents, UDP, and other general protocols, including RTP, RTCP, SRM.

*Links* are modeled either as simplex- or duplex-links with a predefined capacity, delay, and queuing discipline. In addition, links can be torn down or restored at any point in time during the simulation, simulating link failures. Links are built from a sequence of `connectors` objects. The data structure representing a link is composed by a queue of connector objects, its head, the type of link, the ttl (time to live), and an object that processes link drops. `Connectors` receive a packet, perform a function, and send the packet to the next connector or to the drop object. Various kinds of links are supported, e.g. point-to-point, broadcast, *wireless*. The *queues* are considered as part of a link. NS-2 allows the simulation of various queuing and packet scheduling disciplines. Provided C++ classes include: drop-tail (FIFO) queuing, random early detection (RED) buffer management, CBQ (priority and round-robin), weighted fair queuing (WFQ), stochastic fair queuing (SFQ) and deficit round-robin (DRR).

The user has to specify the routing strategy (static, dynamic) and protocol to be used. Supported routing features include asymmetric routing, multi-path routing, link-state and distance vector algorithms, multicast routing, and several ad hoc algorithms. *New protocols* can be added by specifying new agents in C++ and exposing the relevant parameters to the Tcl interpreter. In particular, OTcl is assumed to be advantageous for quick prototyping purposes.

Various types of *applications* can be simulated. Among them are FTP, Telnet, and HTTP, which

use TCP as the underlying transport protocol, and applications requiring a constant bit rate (CBR) traffic pattern, which use the UDP transport protocol.

For the purpose of *traffic generation* NS-2 provides an exponential on/off distribution and it allows also to generate traffic according to a trace file.

*Packet losses* are simulated by buffer overflows in routers, which is also the dominant way packets get lost in the Internet. Other packet losses are related to *error models* where the unit could be packet, bit or time based.

*Arbitrary network topologies*, composed of routers, links and shared media can be defined either by listing the network nodes and edges in a topology file or using some network generators developed for NS-2 (Tiers and GT-ITM).

For collecting output or trace data on a simulation NS-2 uses both *traces*, records of each individual packet as it arrives, departs, or is dropped at a link or queue, and *monitors*, record counts of various interesting quantities such as packet and byte arrivals, departures, etc., that can be associated to both packets and flows. The Network Animator (NAM), is a Tcl/TK based animation tool that can be used for viewing NS-2 trace files for *post-processing, analysis* and *re-play* of simulations (actually NAM can be used with any simulator, as long as data formats are respected).

### Specific support for mobile ad hoc networks

The class `MobileNode` extends the basic capability of the `Node` object class by adding functionalities of a wireless and mobile node like ability to move within a given topology, ability to receive and transmit signals to and from a network interface with an antenna, etc. In addition, a `MobileNode` is not connected by means of `Links` to other nodes or mobile nodes. `MobileNode` is a split object. The mobility features, including node movement, periodic position updates, maintaining topology boundary etc. are implemented in C++, while plumbing of network components within `MobileNode` itself (like `classifiers`, `MAC`, `Channel`, etc.) are implemented in OTcl.

The *network stack* for a mobile node consists of: (i) a *link layer*, (ii) an *address resolution protocol* (ARP) module connected to the link layer, (iii) an *interface priority queue* which gives priority to routing protocol packets and supports running a filter over all packets in the queue, (iv) a *MAC layer* implementing the specifications of the standard IEEE 802.11 (as well as a single-hop preamble-based TDMA MAC protocol), (v) a `Tap Agent` which receives, if allowed, all the packets from the MAC layer before address filtering is done, (vi) a *network interface* which serves as a hardware interface used by the mobile node to access the wireless channel. The network interface is subject to collisions and to the *radio propagation model*, which, in turn, receives the packets transmitted by node interfaces to their wireless channel. The interface stamps each transmitted packet with meta-data related to the transmitting interface, like the transmission power, wavelength etc. This meta-data in the packet header is used in turn by the propagation model in receiving network interface to determine if the packet has minimum power to be received and/or captured and/or detected (carrier sense) by the receiving node. The model approximates the DSSS radio interface (Lucent WaveLan direct-sequence spread-spectrum).

The *radio propagation model* uses a *shadowing model* which, to take into account multi-path propagation effects, represents the received power in terms of a random variable. near distances the

model uses an approximation of the Friss-space attenuation ( $O(1/r^2)$ ), while far distances from the emitting node a Two Ray Ground ( $O(1/r^4)$ ) model is used. The approximations assume specular reflection off a flat ground plane.

*Antennas* used by the mobile nodes are assumed to be omni-directional and with unity gain.

Four ad-hoc *routing protocols* are currently supported: Destination Sequence Distance Vector (DSDV), AODV, DSR, and Temporally ordered Routing Algorithm (TORA). While the Zone Routing Protocol (ZRP) and the Optimized Link-State Routing (OLSR) can be found on the Internet.

Nodes are designed to *move* in a three dimensional topology. However the third dimension is not used, therefore, the nodes are assumed to move always on a flat terrain.

### General summary of NS-2's features

NS-2 is the most popular simulator used in the research field of mobile ad hoc networks. NS-2 comes fully equipped of protocols, models, algorithms and accessory tools, and it is for free. Therefore, in terms of scientific acceptance, number of tools/modules and cost, NS-2 would be a sort of ideal choice.

On the other hand, from the short given description of its architecture, it is quite clear that NS-2 is a rather complex software. Adding new components and/or modifying existing ones necessarily involves writing several software modules (in both C++ and OTcl), taking into account several parameters and multi-step data flows. As a net result, NS-2 is not so easy to use in the BISON perspective of contributing new models, protocols, and studying different scenarios at different levels of detail. Moreover, there is a lack of graphical tools that could greatly help code development.

Unfortunately, the provided documentation does not help from this point of view. In fact, it is often limited and out of date with the current release of the simulator. Most problems must be solved by consulting the highly dynamic newsgroups and browsing the source code.

In general, it is admitted that the *learning curve* for NS-2 is steep and *debugging* is difficult due to the dual C++/OTcl nature of the simulator. In some sense, being NS-2 the result of a rather long process of development, which has incorporated contributions from several different sources, the software design is quite poor with respect to current standards. If it is rather easy to use the simulator, it is not that easy to learn how to add new components or modify existing ones.

Also in terms of (graphical) tools to describe simulation scenarios and analyze or visualize simulation trace files, NS-2 cannot compare positively with commercial tools like OPNET and QualNet.

A more troublesome limitation of NS-2 is its large memory footprint and its lack of scalability as soon as simulations of a few hundred to a few thousand of nodes are undertaken. This is a major impediment according to the BISON's objectives. Only small sized, but at the same time fine-grained simulations can be undertaken.

## 6.4 OMNeT++

OMNeT++ ([whale.hit.bme.hu/omnetpp](http://whale.hit.bme.hu/omnetpp)) is an *open-source*, component-based simulation package built on C++ foundations. It offers a C++ simulation *class library* and GUI support (graphical network editing, animation).

The simulator can be used for: traffic modeling of telecommunication networks, protocol modeling, modeling queuing networks, modeling multiprocessors and other distributed hardware systems, validating hardware architectures, evaluating performance aspects of complex software systems, and, more in general, modeling any other system that can be mapped to active components that communicate by passing messages. In some sense, and differently from the other simulators discussed so far, OMNeT++ is not specifically designed for telecommunication networks, but it is far more general. This fact, together with the fact that OMNeT++ is still a young software product (its development started in 1998), determines that OMNeT++ comes with far less pre-built modules and protocols than the other (network) simulators considered so far. On the other side, OMNeT++ has been carefully designed from the software point of view, resulting in a product which is much better organized, flexible and easy to use than the other simulators, which are all based on older software products and design concepts.

### OMNeT++'s internal architecture

An OMNeT++ model of a system (e.g. a network) consists of *hierarchically nested modules*. The depth of module nesting is not limited, allowing the user to reflect the logical structure of the actual system in the model structure. Modules communicate via *message passing*. Messages can contain arbitrarily complex data structures. Modules can send messages either directly to their destination or along a predefined path, through *gates* and *connections*, which have assigned properties like bandwidth, delay, and error rate. Modules can have parameters which are used to customize module behavior, to create flexible model topologies, and for module communication, as shared variables.

Modules at the lowest level of the module hierarchy are to be provided by the user, and they contain the algorithms in the model. During simulation execution, simple modules appear to run in parallel, since they are implemented as *coroutines* (sometimes termed lightweight processes). To write simple modules, the user is requested to use C++ programming. OMNeT++ has a consistent object-oriented design. One can freely use concepts of object-oriented programming (inheritance, polymorphism etc.) to extend the functionality of the simulator.

OMNeT++ simulations can feature different user interfaces for different purposes: debugging, demonstration and batch execution. *Graphical* user interfaces make the inside of the model visible to the user, allowing him/her to start/stop simulation execution and to intervene by changing variables/objects inside the model. The *GUI library* is linked with the debugging/tracing capability into the simulation executable. This choice enables every user-created object being visible (and modifiable) in the GUI via inspector windows.

OMNeT++ also supports *parallel simulation* through the use of either MPI or PVM3 communication libraries. Moreover, OMNeT++ has also Akaroa<sup>2</sup> support for MRIP, and includes the

---

<sup>2</sup> The Akaroa research project [20] is aimed at improving the credibility of results from quantitative stochastic

same synchronization methods used in PARSEC. In general, OMNeT++ bears some similarities with PARSEC but it is a much more flexible, versatile and rich environment.

It is possible to create any form of *hierarchical topology* by using a human-readable and expressive textual topology description format (the NED language). The same format is used by a graphical editor (GNED). It is also possible to build a network at run-time by program. Both modules and connections among modules can be also dynamically created/destroyed during the simulation run by using library calls.

OMNeT++ comes with an extensive set of container classes for data structures (e.g., queues, arrays), data collection classes, probability and statistics classes (e.g., histograms, exponential distributions) transient detection and result accuracy detection classes. Output files are text files in a format which, after some simple processing, can be read into standard mathematical and statistical packages. In this sense, OMNeT++ does not provide any “duplicate” of such standard packages.

In terms of *contributed models* for telecommunications, OMNeT++ comes with an Internet protocol suite containing IP, TCP, UDP, RTP and some support for basic QoS, while, concerning wireless networks, there is a module for the simulation of lower layers of GSM networks, and some preliminary modules for node mobility, radio propagation, routing algorithms (AODV) and general wireless communication.

### General summary of OMNeT++'s features

The overall design of OMNeT++ is pretty nice and fully based on the driving ideas of object-oriented design. In this respect, OMNeT++ is probably the best software product among the considered ones. The use of the simulator, as well as the definition of new custom models and protocols is quite easy when compared to the other simulators, and to NS-2 in particular. Moreover, it is reasonable to expect a good scalability of OMNeT++, even if there are no real data concerning the simulation of large networks. Therefore, OMNeT++ would perfectly fit BISON's needs for studies at different levels of resolutions and scale, as well as for what concerns rapid prototyping of new models and protocols. Moreover, the nice graphical interface offers good potentialities for monitoring and analysis of the simulations.

Unfortunately, some important network-specific models and protocols are definitely missing. This fact, in spite of some interest that OMNeT++ is widely attracting, dramatically limits the use of OMNeT++. In our case, the use of the current release of OMNeT++ could be limited to very high-level studies and when a quick prototyping of new ideas is required. The choice of OMNeT++ as the primary tool for our studies would necessarily ask for a custom implementation of modules related to the MAC layer, the radio propagation, the application layer, and the node mobility. Moreover, some of the most popular routing algorithms for mobile ad hoc networks (e.g., AODV and DSR) should be definitely implemented.

---

simulation using automated sequential analysis, and speeding up such simulations using Multiple Replications In Parallel (MRIP) to harness the computing power of a network of inexpensive workstations.

## 7 Conclusion: The choice of QualNet as simulation framework

In the light of the review of the candidate network simulators, and according to the previously selected criteria to rank the different simulators, in BISON we have decided to use *QualNet* as the environment for development and simulation. None of the reviewed simulators seem to really possess the whole set of characteristics required by BISON's research plans and objectives. However, *QualNet* appears as the best compromise in terms of number of pre-built components, modularity, scalability, and modifiability. In this sense, we see *QualNet* as an effective simulation framework on top of which we can build the complete BISON's simulation architecture for mobile ad hoc networks, by adding new models and protocols, and by adapting/modifying the existing *QualNet*'s components to BISON's specific needs.

*QualNet* has been chosen since it met most of the requirements discussed in Subsection 5. In particular, it comes with: (i) an extensive set of pre-built models, protocols and algorithms, (ii) a good level of acceptance from the scientific community, (iii) an excellent scalability, (iv) a rather good, highly modular, software design, (v) a satisfactory level of usability, modifiability and expandability, (vi) advanced graphical and mathematical tools for experiment building, monitoring and post-processing, (vii) good documentation, (viii) possibility of parallel and/or distributed implementations, (ix) possibility to specify a realistic 3D model of the environment.

The presence of an extensive set of correct implementations of models, protocols and algorithms will dramatically shorten our startup time, since we will have to worry only about the implementation of our algorithms and of possible modifications/extensions to existing modules. Also OPNET and NS-2 present this important feature, but not OMNeT++.

The scalability up to thousand of nodes will leave us the possibility to carry out studies over a wide range of detail and scenarios. This scalability property and the highly modular nature of the system can in principle allow also statistical-mechanics-like simulations. Among the other considered simulators only OMNeT++ could enjoy this possibility.

In terms of ease to use/modify/extend, *QualNet* seems to be more than satisfactory. In this sense it is probably comparable to OPNET, while OMNeT++, with its good object-oriented design seems to be the best simulator in this respect. NS-2's scores poorly.

The quality of the additional tools for design, monitoring, and analysis is in general at an acceptable level in GloMoSim, and definitely good in *QualNet*. From this point view GloMoSim/*QualNet* compares favorably and/or at the same level with respect to the other simulators.

We plan to start to build our algorithms and simulation architecture by using *QualNet*. In spite of the fact that GloMoSim and *QualNet* have rather similar features and *QualNet* is not cost-free, we have preferred *QualNet* over GloMoSim since *QualNet*'s additional tools for design and analysis seem to be key-features to speedup algorithms' implementation and facilitate the in-depth understanding of their behavior. Moreover, GloMoSim is not anymore supported by its authors, while *QualNet*, being a commercial product, is shipped with a full online support for any sort of technical and implementation issues.

We leave also open the possibility to use either NS-2 or OMNeT++ in the future, if we realize that the use of *QualNet* is in some sense limiting for our research. We also expect that we can migrate our code to GloMoSim with only minor changes (*QualNet* and GloMoSim share

a similar structure, however, function calls might have different names and arguments). More realistically, according to the previous discussions on the problems inherent to the use of simulation and simulators, we might consider the possibility to test our models and algorithms using different simulators in order to carry out analysis at different levels and to obtain stronger statistical validations.

## References

- [1] L. Bajaj, M. Takai, R. Ahuja, R. Bagrodia, and M. Gerla. GloMoSim: A scalable network simulation environment. Technical Report 990027, UCLA Computer Science Department, 13, 1999.
- [2] J. Banks, editor. *Handbook of Simulation : Principles, Methodology, Advances, Applications, and Practice*. Engineering and Management Press, 1998.
- [3] J. Banks. Introduction to simulation. In P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, editors, *Proceedings of the Winter Simulation Conference*, 1999.
- [4] Shared-Cost RTD Project (IST-2001-38923) funded by the Future & Emerging Technologies initiative of the Information Society Technologies Programme of the European Commission, 2003–2006.
- [5] B. Blum, T. He, and Y. Pointurier. Mac layer abstraction for simulation scalability improvements. Technical report, Department of Computer Science, Charlottesville, VA, December 2001.
- [6] T. Camp, 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(5):483–502, 2002.
- [7] D. Cavin, Y. Sasson, and A. Schiper. On the accuracy of MANET simulators. In *Proceedings of the Workshop on Principles of Mobile Computing (POMC'02)*, pages 38–43, Toulouse, France, October 30–31 2002. ACM.
- [8] M. A. Centeno and M. F. Reyes. So you have your model: what to do next. a tutorial on simulation output analysis. In D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, editors, *Proceedings of the Winter Simulation Conference*, 1998.
- [9] DARPA/NIST. DARPA/NIST Network Simulation Validation Workshop. Proceedings at <http://www.dyncorp-is.com/darpa/meetings/nist99may/index.html>, May 1999.
- [10] S. Das, C. Perkins, and E. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *Proceedings of the IEEE Infocom*, pages 3–14, Tel Aviv, Israel, March 2000. IEEE Press.
- [11] J. Heidemann, N. Bulusu, J. Elson, C. Intanagowiwat, 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*, pages 3–11, January 2001.

- [12] J. Heidemann, K. Mills, and S. Kumar. Expanding confidence in network simulation. Technical Report 00-522, USC/Information Sciences Institute, April 2000.
- [13] P. Huang, D. Estrin, and J. Heidemann. Enabling large-scale simulations: selective abstraction approach to the study of multicast protocols. In *Proceedings of the International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 241–248, Montreal, Canada, July 1998. IEEE Press.
- [14] D. B. Johnson. Validation of wireless and mobile network models and simulation. In *Proceedings of DARPA/NIST Network Simulation Validation Workshop*, Fairfax and Virginia, USA, May 1999.  
<http://www.dyncorp-is.com/darpa/meetings/nist99may/index.html>.
- [15] NS-2 Network simulator. <http://www.isi.edu/nsnam/ns/>.
- [16] Object TCL Extensions (OCTL). <http://bmrc.berkeley.edu/research/cmt/cmtdoc/otcl/>.
- [17] OPNET Simulator. OPNET Technologies, Inc., Bethesda, MD, USA. <http://www.opnet.com/>.
- [18] Parallel Simulation Environment for Complex Systems (PARSEC). <http://pcl.cs.ucla.edu/projects/parsec/>.
- [19] K. Pawlikowski, H.-D. J. Jeong, and J.-S. R. Lee. On credibility of simulation studies of telecommunication networks. *IEEE Communications Magazine*, January 2002.
- [20] Project Akaroa. [http://www.cosc.canterbury.ac.nz/research/RG/net\\_sim/simulation\\_group/akaroa/about.shtml](http://www.cosc.canterbury.ac.nz/research/RG/net_sim/simulation_group/akaroa/about.shtml).
- [21] QualNet Simulator, Version 3.6. Scalable Network Technologies, Inc., Culver City, CA, USA, 2003. <http://www.scalable-networks.com>.
- [22] REAL Network simulator. 1997. <http://www.cs.cornell.edu/skeshav/real/overview.html>.
- [23] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, 1999.
- [24] R. Y. Rubinstein. *Simulation and the Monte Carlo Method*. John Wiley & Sons, 1981.
- [25] Y. C. Tay and K. C. Chua. A capacity analysis for the IEEE 802.11 MAC protocol. *ACM/Baltzer Wireless Networks*, 7(2):159–171, March 2001.
- [26] Virtual InterNetwork Testbed (VINT). 1998. <http://www.isi.edu/nsnam/vint/index.html>.
- [27] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: A library for parallel simulation of large-scale wireless networks. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulation*, pages 154–161, Banff, Alberta, Canada, May, 26–29 1998.