A Flexible Visual Simulator for Wireless Ad-Hoc Networks of Mobile Nodes

Tullio Facchinetti
University of Pavia
Pavia, Italy
tullio.facchinetti@unipv.it

Giorgio Buttazzo
University of Pavia
Pavia, Italy
buttazzo@unipv.it

Luis Almeida
DET / IEETA - University of Aveiro
Aveiro, Portugal
lda@det.ua.pt

Abstract

The management of ad-hoc networks raises interesting problems, that are particularly challenging for networks of mobile nodes. Considering the inherent complexity of these systems, the development of distributed applications relying on wireless communication protocols would be greatly simplified by the use of specific tools for supporting testing and step-by-step debugging.

In this paper we describe WISE, a flexible interactive simulation environment for the development of wireless ad-hoc networks consisting of mobile units. A graphical interface allows the user to create/delete nodes, change their positions and parameters, and select specific mobility models in order to verify the network behavior in dynamic conditions. The simulator also provides a useful support for the verification of agreement protocols, synchronization algorithms and distributed scheduling, allowing the user to display a step-by-step evolution of the algorithms in a suitable graphical representation.

Keywords: wireless networks, simulation environment, mobility.

1 Introduction

Recent advances in wireless communication technology and a progressive reduction in the associated costs are boosting the use of wireless networks in many diverse domains, either to integrate office equipment, personal equipment, or even to interconnect sensory and actuating devices.

One of the fields in which there is strong interest in the use of wireless networks is the interconnection of mobile units with sensing, processing, and communication capabilities for monitoring and exploration purposes.

In some cases, the possibility of controlling the position of the nodes (through teleoperated or autonomous mobile robots) would allow much higher flexibility, because the network could be configured for a better coverage of the sensed area, or for actively following the evolution of the phenomena [9, 12]. When nodes move, however, they need to interact with the environment (e.g., to avoid obstacles) and with the other nodes of the network; hence, most of the activities carried out by the team need to be executed under timing constraints, that must be enforced on tasks to guarantee a minimum level of performance. In some applications the communication among distant units is based on a wired backbone. However, in the considered above situations, a wired infrastructure cannot be used, hence a full autonomy of the team units can only be achieved using an ad-hoc network.

Design, analysis and test of mobile ad-hoc wireless networks take great advantages from specific tools. Although some basic network and communication protocol properties can be theoretically proved, in other cases the networks performance and their behavior in particular conditions can only be tested by simulation. In particular, this is true for networks made by mobile units, since several different topology configurations are possible and the network connectivity changes dynamically. Moreover, the analysis of MANETs include several issues: area coverage, localization, connectivity, message routing, and timeliness in message exchanging [13].

Several simulation environments are currently available in the academic world. The Network Simulator ns-2 [1] is one of the most widely used tool for the performance analysis and evaluation of network protocols. It recently included features to simulate wireless networks with the introduction of the MobileNode class. However, it cannot be used for online demonstrations and interactive debugging, since it does not provide online graphical output. The Cnet simulator [11] is an easy-to-use simulation tool. It implements the ISO/OSI stack paradigm and can simulate both Ethernet based LANs and token ring based LANs, but it does not support wireless communication and does not provide online graphical output. Netsim [5] is a powerful simulation tool, but it does not support wireless networks, neither a graphical interface. The GloMoSim [4] simulation environment was developed for simulating large-scale wireless networks: it is multi-platform and provides a graphical interface, but it requires the use of the PARSEC environment [2] as a parallel simulation engine to provide high computational power.

This paper describes WISE, Wireless Interactive Simulation Environment. WISE is a flexible visual simula-
tor specifically developed for ad-hoc networks of mobile units. The simulator provides a basic infrastructure to simplify both the implementation and debug of new algorithms. WISE has been designed to be user-friendly and can be used to analyze a protocol and to present a graphic step-by-step evolution of the algorithm behavior. For this purpose the package combines the simulation engine with a highly customizable graphical interface.

The rest of the paper is organized as follows: Section 2 presents the general architecture of the simulator; Section 3 describes the most relevant implementation details; Section 4 illustrates the the operational modes of WISE; Section 5 introduces the topics related with the graphic engine; Section 6 provides an example of usage; finally, Section 7 states our conclusions and future work.

2 The simulator

WISE has been designed to simulate distributed protocols for wireless ad-hoc networks consisting of mobile nodes. The main features of the simulator include:

- Native support for node mobility, to test the behavior of the network in dynamic environments;
- Modular software structure;
- Object-oriented programming;
- Interactive graphical interface to simplify the design of a network and the verification of network protocols.

Figure 1 illustrates the hierarchical model adopted for WISE. The model is node-centric, meaning that the Node is the elementary component of the system. Nodes can be grouped into one or more Networks, fully interacting among themselves. The simulator also supports a modeling of the Environment where the nodes move.

Each hierarchical level is characterized by specific properties: nodes are characterized by their position, individual behavior (i.e., moving strategy), transmission power, etc. Networks may have a maximum number of nodes, positioning limitations, and other constraints. The Environment may introduce obstacles, sources of interference and noise.

The elements of a level may affect the properties of the elements in another level. For example, the network topology is influenced by the nodes position; its connectivity is affected by the presence of obstacles and by the environmental interferences, which may disturb the communication between two nodes. Actually, the interaction between a Node and the Environment is unidirectional, since the Environment may affect the behavior of the Node through obstacles and interferences, but the Node cannot affect the Environment.

Nodes can be freely inserted in the environment, and they can be manually moved during the simulation in order to test particular topological configurations of the network. A manual control of the node position during a step-by-step session can be very useful to verify the effects of hidden terminals on the behavior of the tested algorithm.

3 Implementation of the model

The implementation WISE aims at achieving two levels of flexibility:

1. portability of the kernel on different operating systems;
2. independence among different logical modules.

The WISE core is implemented in the C++ language. On one hand, it ensures high efficiency and strong optimizations in the WISE simulation core; on the other hand, it achieves good portability between different operating systems. However, portability problems may arise from the graphical interface: typically, the graphic engine is strongly operating system dependent.

In our implementation, the simulator kernel is separated from the graphical interface, thus allowing the simulation even in the absence of graphical support. For example, the simulation results may be collected in a text file to be used later for off-line analysis with specific tools. An additional advantage of separating the simulator kernel from the graphical interface is the fact that the same simulation results can be represented under different views, thus making their interpretation easier and immediate. For example, two different windows may be used to separate different representations of the same data and a third window may compare the two representations.

The kernel has been implemented by taking full advantage from the object oriented hierarchical paradigm provided by the C++ language. At the bottom of the hierarchy there is the TBasicNode class, that forms the data background common to almost all wireless environments, mainly concerning the node position. The TDdrawNode class extends the basic node representation by adding few properties needed to simplify the graphic node representation. Other classes are derived from TBasicNode and TDdrawNode in order to represent specific types of nodes. They just add the data structures and functions needed for specialized simulations, like transmission ranges and channel frequencies.

Every node is a member of a network. From the implementation point of view, this means that the TNetwork class is a collector of TBasicNode objects. In the same way, TEnvironment collects a set of TNetwork classes, since all the networks reside in the environment. New nodes are added to existing networks, and new networks may be added to the environment. The TNetwork and TEnvironment containers provide the functions to deal
with their members. For example, \textit{TEnvironment} provide a function to evaluate the distance between two nodes and this function is used to check whether a node is able to listen to another node transmission.

Whenever a node with a new behavior is needed, the class hierarchy has to be extended to derive the new element from the \textit{TBasicNode} class. For example, a node moving accordingly with a specific mobility model inherits the characteristics from a common node, but implements the new behavior for the moving strategy.

4 Running a simulation

The simulation goes on as a potentially infinite series of steps, called \textit{ticks}. At each tick, an event is generated within the simulation environment, and every component executes the corresponding action. For example, a node may update its current position according to the mobility model parameters, and/or it may need to broadcast a message acting upon its communication algorithm.

WISE can run a simulation under three modes:

- in \textit{step-by-step} mode;
- in \textit{live} mode;
- in \textit{batch} mode with (optional) online graphical output.

The three simulation modes depend on how the ticks are generated: in \textit{step-by-step} mode, each tick is generated after an explicit command issued by the user; in \textit{live} mode, the ticks are triggered periodically by an internal timer with selectable period; finally, in \textit{batch} mode, each event associated with a tick is forthwith generated after the end of the computation started by the previous tick. The last mode is the fastest and may not retrieve a graphic feedback.

The three options have been proposed for different purposes. The step-by-step update is useful to tune the system parameters and to debug the algorithm. The periodic update may be used to monitor the online system behavior, by observing the dynamics in link establishment or breaking together with the system data flow. The batch mode can be used to perform simulations in which the output data are made available for an off-line evaluation.

WISE is expressly developed to simulate networks of mobile nodes. It actually implements the Gauss-Markov mobility model [10] as well as the Random Waypoint mobility model [6]. Moreover, the \textit{step-by-step} mode allows the user to manually implement an adversary-based mobility strategy [3] and other models based on the creation/deletion of specific links among the nodes [8, 14]. This means that the user can opportunistically change the network connectivity during the simulation to easily build specific test case situations.

The user can switch between the step-by-step mode and the live mode at run-time. This is useful to let the system run until a specific condition is reached and then to analyze the system behavior more in details.

5 Interactive graphic interface

A good graphic feedback is one of the primary goals of WISE. It may help to better understand the algorithm behaviors and gives an appealing representation of protocol data flow and network performance. This may be used to show complex situations in a real-time fashion, and it is highly desirable in order to make live presentations, i.e for teaching purposes.

The visual interface is made by a main window (shown in Figure 2) which displays the graphic overview of the network deployment, with options to show or hide links among the nodes and transmission ranges (both options are set in the example of Figure 2). By using the mouse, each node can be freely moved in order to obtain the desired network configuration, establishing new links or removing existing ones. New nodes can be added in just a mouse click. The application also supplies a set of child windows, each one related to a specific simulation module.

Figure 3 shows the relationships among the windows that compose the application. A number of child windows (the \textit{simulation modules}) display the evolution of the protocol data flow and every simulation module may have a related window containing the required optional parameters.

During the simulation in \textit{step-by-step} and \textit{live} mode,
the network topology represented within the main window is updated after every tick. While a step-by-step simulation is running, the nodes can be moved and the node parameters may be changed to simulate the desired situation.

6 Example of usage

The WISE simulator has been profitably used to derive many properties of the MAC level communication protocol presented in [7]. WISE has been used to optimize the algorithm and verify the convergence. Moreover, it has been helpful to study the dynamic behavior of the method by manually moving the nodes in order to change the network topology configuration during step-by-step simulation. This feature of the simulator has been particularly useful to investigate worst-case topologies and to understand the influence of link establishment/disruption on the communication system.

Figure 4 shows the output produced by the Topology Simulation window, which displays the evolution of an algorithm for the distributed network topology reconstruction. In this simulation, the nodes exchange periodic messages about their own view of the global network topology. After every transmission, such an information is used by the nodes which receive the data to update their own state, making all the nodes views iteratively converge to the correct overall network topology. Figure 4 shows the last three steps of the topology matrix updating process, where the matrix at \( i \)-th row and \( j \)-th column represents the matrix owned by node \( i \) at the \( j \)-th simulation step. The highlighted matrices identify those corresponding to the correct global topology configuration, to simplify the identification of the converged matrices. The same figure also shows the window containing both the simulation and visualization options.

7 Conclusions

In this paper we discussed the importance of using appropriate simulation tools for the development, maintenance, and testing of new distributed algorithms for wireless ad-hoc networks of mobile nodes.

The solution proposed in this work is WISE, a flexible visual simulator specifically developed to interactively display the evolution of wireless mobile networks. The simulator natively integrates the most common used models to describe the node mobility. One of its primary goals is a good visual feedback to simplify the comprehension of the problem under analysis. Thanks to its user-friendly graphical interface, WISE can be effectively used to make live presentations, i.e. for teaching purposes.

References