An interactive VHDL simulator for IEEE 802.11 networks

F. BELLOTTI, A. DE GLORIA, D. GROSSO, L. NOLI and M.OLIVIERI
Department of Biophysical and Electronic Engineering (DIBE)
University of Genoa
Via Opera Pia 11a 16145 Genoa
ITALY

Abstract: - In this paper we present a user-friendly simulator for IEEE 802.11 networks. The simulator is composed of an interactive Java-based graphical interface and of a detailed VHDL model, running on a VHDL simulation engine. In particular our VHDL description include the Medium Access Layer, that is the core of the protocol functionality and performance, and the physical layer. The use of our model allows the analyzer to get precise and reliable evaluation of the protocol performance and to realize a sort of network monitoring by obtaining important information such as the collision rate and the channel occupation of each station. Thanks to the interactive graphical interface our tool is suitable for utilization by hardware designers, communication systems specialists and telecommunications students as well. Particular attention is spent to model typical situations of the transmissions over the wireless channel such as different delays between the stations to simulate the effect of the distance between them, or the presence of hidden terminals or the effects of the collisions. Finally we report the results, in terms of throughput and transmission delay, that we obtained by simulating a network with eight stations. CSCC'99 Proceedings, Pages: 2331-2339

Key-Words: - IEEE 802.11, wireless networks, transmission protocols, Java, interactive simulator, education.

1. Introduction

IEEE 802.11 [6] is one of the most widely accepted standards for *wireless local area networks* (WLANs). WLANs have gained strong popularity in many industrial and business markets because of the possibility of using hand-held terminals to transmit real-time information to centralized hosts for processing. In addition, thanks to wireless technology, users can access shared information from anywhere in their organization without being connected to any wired network. Thanks to the above reasons today WLANs are recognized as a general-purpose data communication system alternative to wired LANs.

The IEEE 802.11 standard defines the medium access control (*MAC*) layer and the physical (*PHY*) layer, compatible with the existing standards for higher layers (link and higher) [4], of stations suitable to be connected to a wireless 802.11 network The project 802.11 supports data rate of 1 Mbps and 2Mbps; the operating frequency that has been allocated for the 802.11 WLANs ranges from 2.4 to 2.8 GHz. Currently WLANs equipment is based on numerous proprietary technologies which do not work with other vendors' equipment. The aim of the project 802.11 is to built a universal standard for the wireless communication in such a way that a buyers should be able to purchase products from any number of vendors without fear of incompatibility.

In this paper we present a VHDL simulator of a IEEE 802.11 based wireless network.. The base node of the network consists of an exhaustive description of the MAC and of the physical layer. This description covers all the MAC functionalities and services and represents a physical layer able to supports both the two prevailing spread spectrum technologies: direct sequence spread spectrum (DSSS) [1,9] and frequency hopping spread spectrum (FHSS) [11,12].

The simulator allows the user to build and to configure a wireless network by varying the number of its nodes and the value of their main parameters.

The simulator is composed of an interactive graphical interface and of a VHDL entity.

The graphics interface has been developed essentially to let the access to the configuration model and to the simulation results available also to people who is not necessarily acquainted with the complex world of a hardware system level simulator . This feature may be particularlty attractive mostly for two kinds of applications. First, it makes possible to augment the level of control over the network project, in the early phase of development, also by communication systems experts, who may not be hardware specialists. Moreover it makes the simulation environment available to students for educational purposes, without requiring any specific knowledge of the internal functionality and structure of the simulator. In fact the

technical background required to cope with the interface exclusively consists in the knowledge of some concepts about the telecommunication systems, as will be explained in the next sections.
Through this model we simulated the entire MAC functionalities and services. obtaining reliable evaluations of the protocol performance. Our work was developed starting with a large interpreting effort of the standard specification [6]; this document is aimed at describing the standard and all its features without any implementation suggestions. In this paper we present the structure of our simulator and the obtained performance measures.

2. Technical background

A wireless local area network is a flexible data communication system that uses radio frequency technology to receive and transmit data over the air. The IEEE 802.11 defines a basic architecture for wireless networks. They are composed of basic service sets (BSSs) each of them connected with a distribution system (DS). The DS is not specified in the standard [6], but is intended to be the means (which is usually wired) through which different BSSs can communicate. Each BSS consists of a group of mobile stations associated to a single server called access point (*AP*). An *AP* is a node that implements both the 802 and the DS protocol and can connect the stations of its BSS to the external word. The set of BSSs interconnected by the DS forms an extended service set (ESS).
The basic access method (referred to as distributed coordination function) implemented by the 802.11 stations, is the *carrier sense multiple access* with *collision avoidance* (*CSMA/CA*) [2,8,10]. This method implements the listen before talk mechanism. This means that a station having a frame to transmit must first sense the state of the channel in order to determine if another station is transmitting. If the channel remains idle for a fixed period of time (called *DIFS* [6]) the station can proceed with its transmission. If the medium is busy the station begins the *backoff* procedure that consists of waiting for a random period of time before listening again the channel. If the channel is still busy another *backoff* interval is selected that is shorter than the previous one. This process is repeated until the station is allowed to transmit. The *backoff* procedure minimizes collisions during contentions between multiple stations.
In order to solve the problem of hidden terminals the protocol provides an alternative way of transmitting data. When the station get the channel by following the *CSMA/CA* procedure it sends a short frame, called request to send (*RTS*) [6,14]; on receiving an *RTS* the receiver answers with a clear to send (*CTS*) frame [6,14] after a period of time, called *SIFS*, shorter than a *DIFS*. The *CTS* signals to the transmitter the successful reception of the request. The reception of the *CTS* enables the transmitter to send the data frame after waiting for a *SIFS*. Both the *CTS* and the *RTS* contain the duration of the subsequent transmission; thus, if all the stations of the *BSS* can hear either the *CTS* or the *RTS* a possible conflict is avoided.
In the following sections we present the structure of the simulator and all its components.

3. The Simulator

The purpose of our work was to build a flexible and user friendly tool for the simulation of a configurable BSS. This goal was reached by providing the environment where any number of stations can operate. Each station consists of an architectural component with a structural and/or behavioral VHDL description. Through the simulator the user can select and introduce operative conditions that affect the transmission of data on the wireless medium. For example it is possible to introduce different delays between the stations to simulate the effect of the distance between them, or to introduce hidden terminals or to consider the effects of the collisions. By simulating our model in function of the medium operative conditions and of some typical features of the 802.11 stations (that we will see in the next section) we obtained evaluations of parameters that characterize the protocol performance: the *throughput* and the *transmission delay*. With the expression transmission delay we indicate the average time between the moment the frame is scheduled to be transmitted and the moment the frame is received. This means that the time spent to access the bus is included in the *transmission delay*.
The above consideration suggests that the model we built can be used as an emulator of the protocol. We chose the VHDL language to build the simulator .The use of VHDL is motivated by its wide usage and by the fact that it can be used in all design phases, from documentation to simulation and synthesys. Our simulator is not only a way to test an access procedure: it is a tool to verify if the model of station to be synthesised works as expected, and if the network built with our station model gets the same statistic results that characterise the protocol. Fig. 1 shows the logic structure of the simulator. The stations, connected through the *channel simulator*, communicate with a sort of off-line link layer controller (*LLC*) that provides the stimuli and records indications and confirmations that the *MAC* layer sends to the *LLC* [7]. In order to provide the required results the *interpreter* gathers and processes some particular data generated by each station. The tasks of the *LLC* are performed by a

module that we called *genesis*. This module is the source of data, instructions and parameters that will determine the station behavior during the simulation. The genesis module takes as input the indications of the graphical interface through which the user can set the charachteristics of the network in terms of topology (number of stations and presence of hidden terminals), trasmission strategy and load desired on the wireless medium. The transmission strategy depends on the value of some of the *MIB* (Management Information Base) parameters [6], that the user can set through the interface. The *MIB* parameters are a great number of variables that define the state of the station, heavily affecting its behavior. For the traffic we decided to use the load model called *buffered load* as shown in [3] The *buffered load* represents the mean amount of data that are waiting to be transmitted, and it equals the throughput we would obtain if the time spent in the access procedure was zero. The *buffered loads* of the entire network is the sum of the *buffered load* of all the active stations within the network.

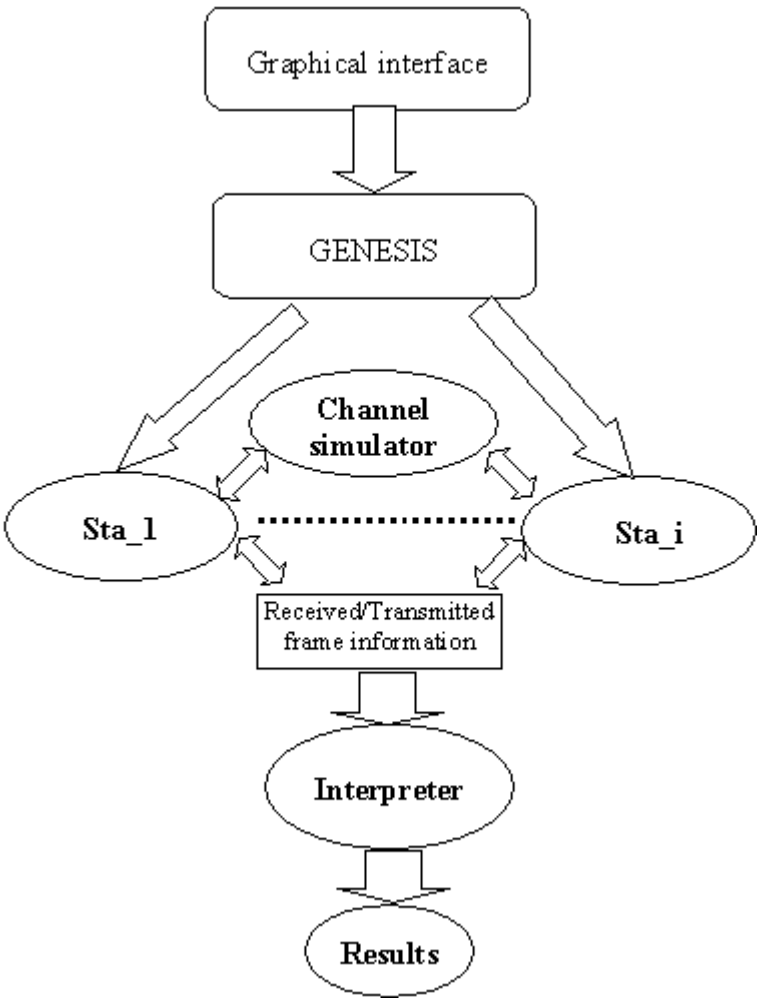

Figure 1The logical structure of the simulator

The genesis module builds the user defined network an generates the *MAC* instructions for the stations. In the same manner, the data to be transmitted, are turned into *MAC* frame and then passed to the stations. The channel simulator models the wireless media by interconnecting the *tx* and *rx* channels of all the stations. By exploiting these connections it inserts delays between stations, simulates the presence of hidden terminals and manages the collisions. In addition in the next section we will see that a probabilistic method has been developed to simulate the presence of noise on the wireless channel.

When the simulation starts, a reset signal is sent to all the stations in order to awake them. During the simulation every station yields information to track the history of the simulation. In particular every station produces information about received/transmitted frames that includes the transmitter/receiver address, the reception/transmission instant and the time of the frame scheduling. Furthermore the duration of the simulation is recorded. The interpreter uses these data to get the performance measures in terms of *throughput* and *transmission delay*. Finally, in order to simulate the data exchange between *MAC* and *LLC*, every station

passes to the genesismodule the received frame, the information about the transmitter address and the result (success or failure) of each executed instruction

4. The station

The stations included in our *BSS* model consists of a *MAC* layer and a physical layer (*PHY*).
Our *MAC* layer is capable of supporting both the *DSSS* and the *FHSS* Physical layer, and a transmission rate of 1 or 2 Mbps. According to the standard specification [6], by setting some *MIB* parameters it is possible to enable the *MAC* to use the *RTS/CTS* exchange or the fragmentation [6] in data transmission.
Each station takes as input the reception channel and the reset signal and its output are the transmission channel and a *data_end* signal which is set after the station executes its last instruction.

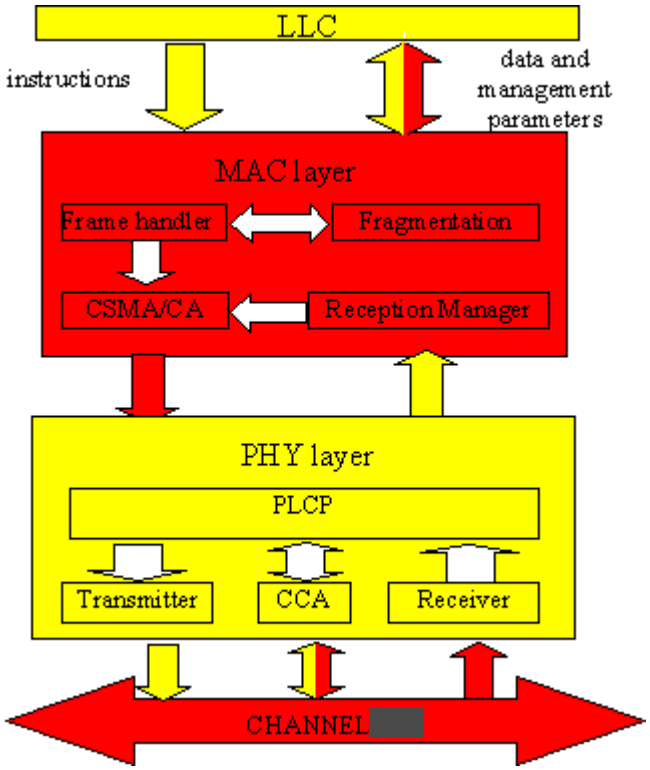

Figure 2 Architectural scheme of the station.

The channels are composed of a data field, used to transmit the packets, and a state field set to busy when a station begins its transmissions.
The architectural scheme of a station is shown in Fig. 2. Our IEEE 802.11 station model is composed of a set of concurrent processes, and it is split in two main parts: transmission and reception part. Each process realizes one or more 802.11 functions and exchanges information with other processes to know the station state. The *MAC* activity starts either when a new *LLC* instruction has to be executed or when a frame has been received by the *PHY*. In the first case the *instruction manager* turns the *LLC* command into a *MAC* format. Each instruction contains parameters necessary for its execution; for example a *send_data* instruction contains the type of frame to be transmitted, the destination address and the access procedure. After instruction decoding, the *frame handler* builds the frame according to LLC directives and the state of *MIB* parameters. For example the *MAC* can decide to use the fragmentation and the *RTS/CTS* exchange depending on the frame length and of the value on particular *MIB* parameters. At this point, if required by the instruction, the execution of the access procedure takes place. At first, the *MAC* layer gets information about channel state through the two supported ways: the virtual and the physical mode [6]. The physical mode in 802.11 stations is implemented through the standard *clear channel assessment* (*CCA*) [6] procedure; in our case the physical procedure is realized with a test of the channel state field. In order to simulate the presence of noise on the wireless channel we implemented a probabilistic *CCA*; this means that our *CCA* procedure makes errors with an a-priori fixed probability.
The virtual mode exploits the duration field contained in each transmitted frame. This field communicates the length of the actual transmission to all the stations. Each station records the value of the duration field in the network allocation vector (*NAV*) [6] and decrements it; the virtual procedure sees the channel free when the

NAV contains the zero value.
If both the virtual and the physical procedure sense the channel idle for more than one *DIFS* a transmission starts immediately, else the *backoff*-procedure begins according to the standard. Once the station gets the channel through the access procedure, the scheduled frame is transmitted to the physical layer. At this pint the *physical layer convergence protocol (PLPC)* [6] sub-layer completes the *MAC* frame with the physical preamble, (*DFSS* or *FHSS*) and the *physical medium dependent* (*PMD*) [6] sub-layer puts the frame on the channel and sets to busy the channel state.
When the station is not physically transmitting, it always listens to the channel, waiting for frames transmitted by other stations.

Network topology parameters	
Number of stations in the network	An integer ranging from 2 to 12
Hidden terminals	The couples of terminals that cannot communicate between each other
Workload definition	
Buffered load	Expressed as Kbytes/sec. A floating point number
Buffered load step	Is the interval of Kbytes/sec through which the simulator steps if the user requires the WIA mode
Strategy of transmission	
Number of packets	Overall number of packets to be transmitted by each station
Length of packets	Length in bytes of the high level messages to be transmitted
MIB parameters	
CTS/RTS Threshold	An integer defining the threshold over which the CTS/RTS transmission mode is triggered
Fragmentation Threshold	An integer defining the maximum size over which packets have to be fragmented for delivery

Table 1 The meaning of the data provided by the user through the interface.

When the antenna captures a frame the *PLCP* sub-layer verifies its correctness and sends it to the *MAC* layer without physical preamble. The *MAC* device called *reception manager* starts analyzing the meaning of the received frame. If it is not addressed to the station the *reception manager* decodes the duration field and updates the NAV, else it decodes the entire frame and sends information to *LLC* about what it contains and activates the suitable actions. For example, if the received frame is an *RTS* frame the *reception manager* after a *SIFS* answers with a *CTS* frame.

5. The graphics interface

While the network communication system simulator can be accessed by the developers of the model or by people discretely acquainted with the VHDL language, a good enhancement to the package consists in the possibility of exploiting a graphics interface through which end users can interact with the complex simulation world using “their own words”. The complexity of the simulation environment, which is essentially thought for the interaction with expert programmers, could prevent potentially interested users from using it, even if a small set of features of the tool are to be exploited and known in this last case.
The graphics interface is of particular appeal if we consider the actual utilisation of the tool by people who is interested in the results of the simulation. In fact the network simulator is likely developed by hardware specialists while the performance of the overall system is to be analyzed by communication systems experts. And these kinds of knowledge are not always mastered by the same person. Our tool makes it easier for technical people to communicate, allowing a better sharing of the competences and a deeper control over the project in all the development phases. Moreover, the presence of a graphics interface makes this tool particularly suitable for an utilisation in the educational field. Students and almost newbies can see an application of the theoretic concepts and gain familiarity with the numbers, the quantities and the problems that the real communication systems involve.

The lexical terms through which users can interact with the interface are those typical of the communication systems jargon, which have been presented in the technical background section of this paper.

The input interface

The input interface collects user provided data about the quantities reported on table 1. We must highlight that the present release of the graphics interface does not provide access to all the results and all the functionalities of the underlying simulator. In phase of presentation of the input interface it must also be underlined that the tool can be used to evaluate a single configuration of workload (Normal mode) or to anlyse the impact of different workloads on the same system (WIA mode, Workload Impact Analyzer). Fig. 4 represent a snapshot of the interface.

The output interface

In the normal mode the interface shows the average simulation obtained values for the system throughput and for the transmission delay. In the WIA mode it outputs two graphs that plot the throughputs and the delays corresponding to buffered loads equispatieted of intervals of Kbytes/sec desired by the user. Fig 5 illustrates an example of results obtained in the WIA mode.

Development considerations

The interface has been written in Java [14], because of the relatively ease of development of visual interfaces in this language and because of the portability and security features assured by Java bytecodes [13, 15]. In fact, though we are running the interface code on the same Pc hosting the VHDL simulator at present, it has already been forseen the possibilty of decoupling the applications. The interface could in fact be embedded as an applet in a html page and deployed over the Internet, dramatically enlarging the set of the potential users.

Future enhancements to the interface may involve, essentially for the educational sake, the accurate visualisation of the different processing phases through which an high level application packet is elaborated during its path along the MAC and the physical layer. In fact, not only is the original packet combined with Correction Redundancy Code (CRC) at both the levels, but also information about source, destination and estimated transmission time is added by the MAC, while the physical layer adds the preambole for the synchronization of the demodulator and other data describing the length of the paket and the transmission bitrate.

6. Simulation Results

In this section we show the results obtained by simulating the model of a network with eight stations. Results are expressed in terms of *throughput* and *transmission delay*. We computed the *throughput* as the total number of correctly received bytes divided by the simulation time; for the transmission *delay* we considered the time between the instant in which the frame begins the access procedure and the instant in which the frame is correctly received by the destination station. All of our simulations are realized as a function of the user defined *buffered load*. Once the user sets the value of the load expressed in Kbytes/sec, the *genesis* module generates it by providing a sequence of *send_data* and *wait* instructions

We realize simulations with and without hidden terminals. We simulated the presence of hidden terminals by setting that station n. 8 cannot listen to stations n. 3, 4 and 5 In this case it is possible that a station senses the channel idle even if another station is transmitting causing a collision.

First we consider the situation without hidden terminals. In case we use the *CTS/RTS* exchange before transmitting the data packet we obtain that the throughput (Fig. 3a) grows with the packet size. The increase is smaller for big packet size because the *CTS/RTS* length becomes negligible with respect to the data length. For a low buffered load the channel is free and the *transmission delay* (Fig. 3c) consists of the time spent to put the frame on the channel. For a high buffered load the channel is busy and the delay grows with the packet size because the *backoff* procedure becomes longer. In case we do not use the *RTS/CTS* exchange, we have higher performance for low packet size as shown in Fig. 3b and 3d. For high packet size the time spent for re-transmission of lost data due to a collision is greater than the time spent to transmit the *CTS/RTS* exchange. Thus we have a loss in performance. Anyway, as collisions are not very likely, the difference between the two previous situations is not great. If hidden terminals exist the use of the *CTS/RTS* frames leads to a great gain in terms of *throughput* and *transmission delay* as shown in Fig. 3e, 3f, 3g and 3h. Finally we present the results obtained by introducing a probabilistic *CCA* procedure. This consists of considering a fixed error probability during the physical channel sensing. This simulates the error introduced, in the real situation, by the presence of noise, of the capture effect and of fading. We realized with a fixed packet dimension (1024 byte) and by varying the error probability. Fig. 3i and 3l show how this kind of errors can dramatically affect the performance of the wireless network.

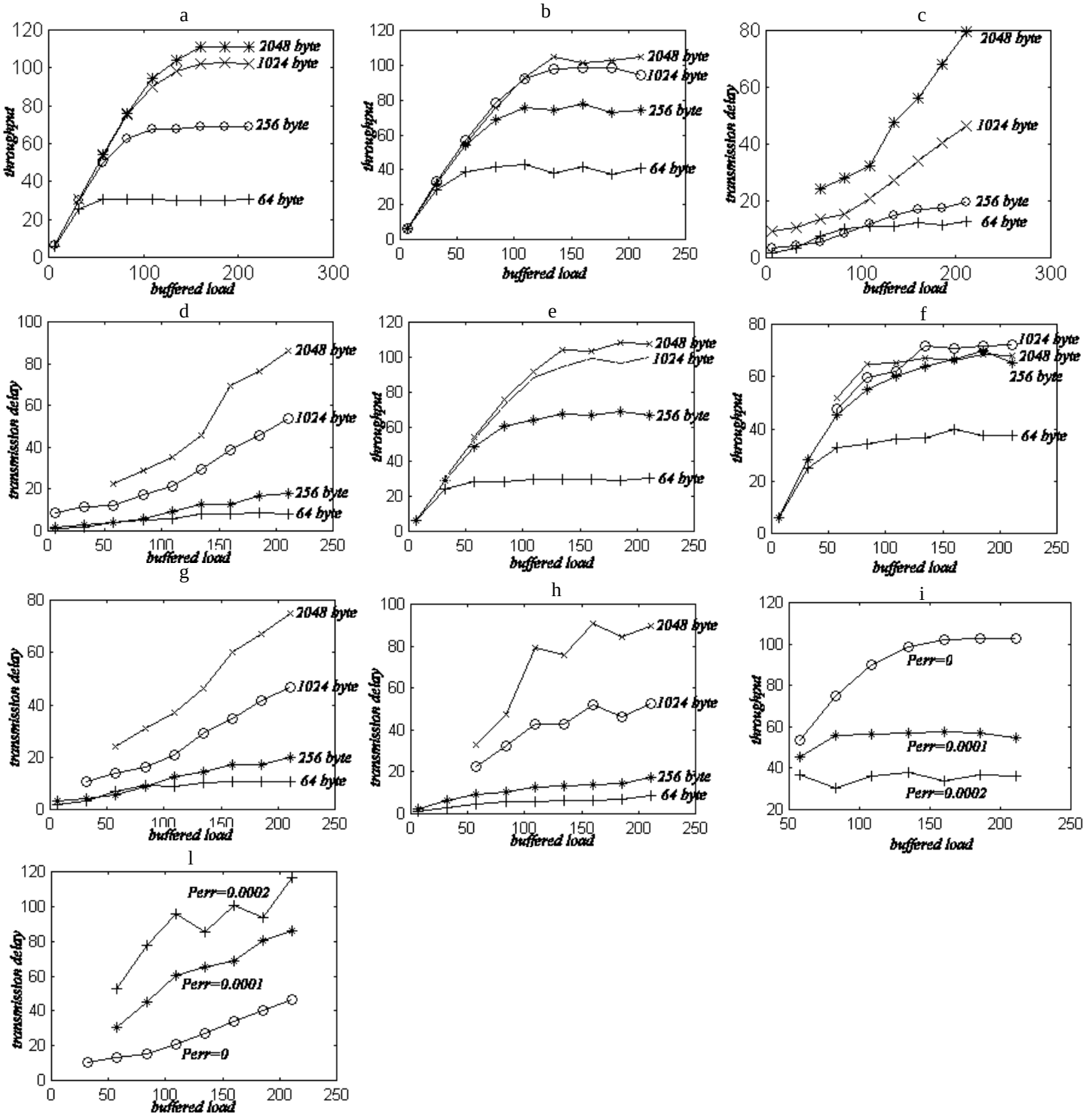

Figure 3 Simulations results. Network without hidden terminals: Throughput with (a) and without (b) cts/rts exchange; transmission delay with (c) and without (d) cts/rts exchange. Network with hidden terminals: Throughput with (e) and without (f) cts/rts exchange; transmission delay with (g) and without (h) cts/rts exchange. Network with probabilistic CCA; throughtput (i) and transmission delay (l)

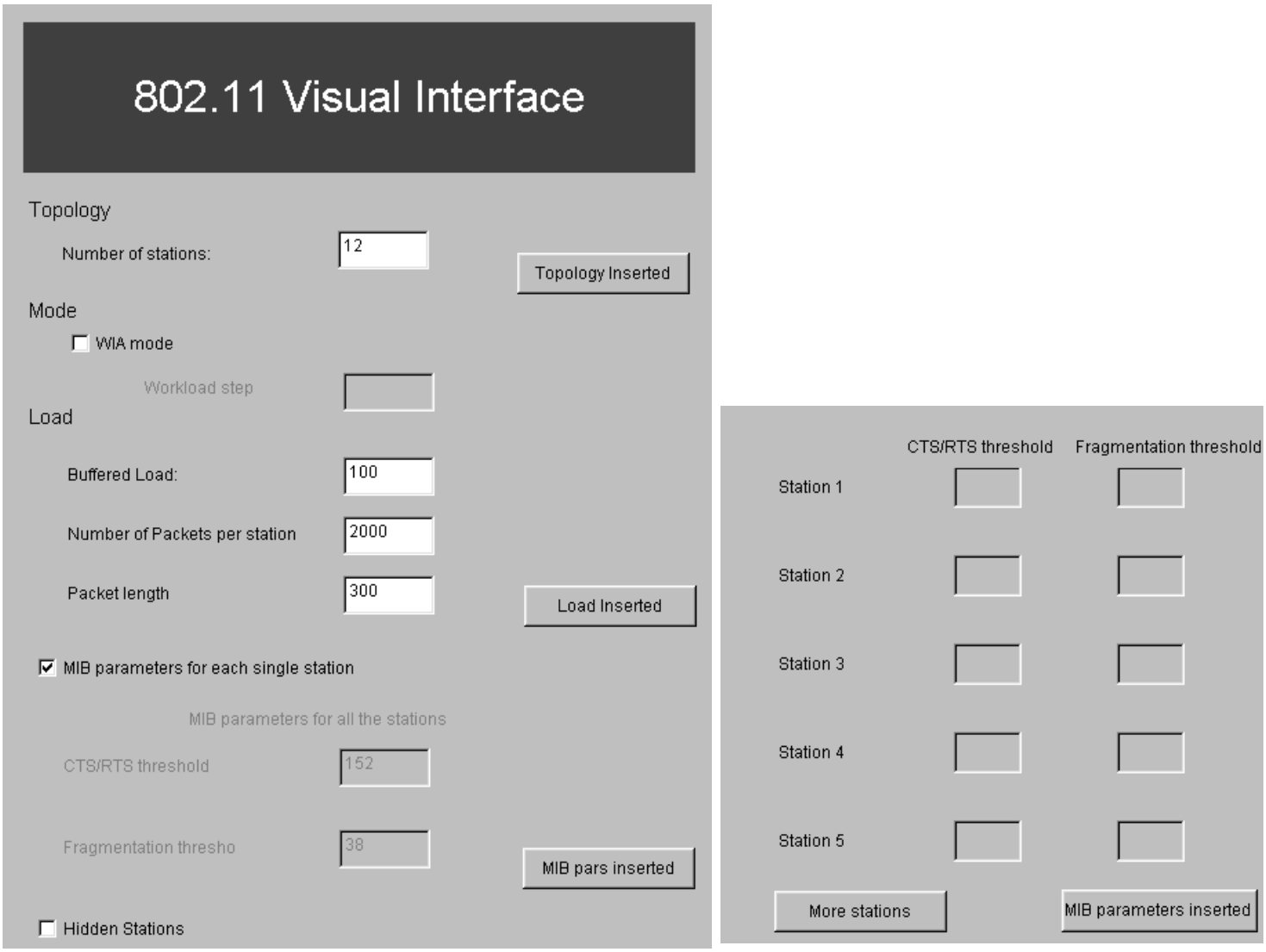

Figure 4 Snapshots of input interface

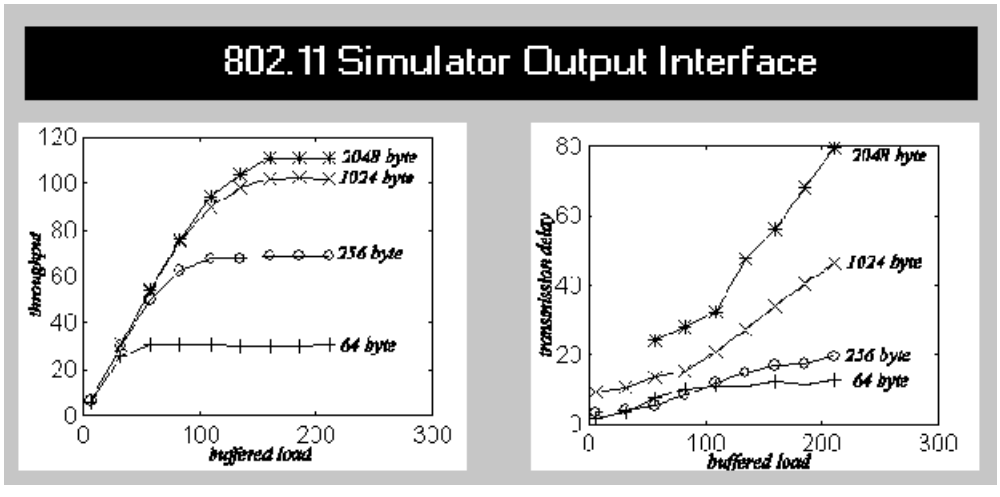

Figure 5 Snapshot of output interface

7. Conclusions

We have presented a simulation environment for evaluating wireless network configuration. The result of this work is a valuable tool for wireless network designers, as it offers a user-friendly Java-based graphic interface and implements a sophisticated VHDL model of the whole system. The work is also an interesting example of the concurrent execution of Java programs and VHDL model simualtion.

References

[1]. C. Andren, "A Comparison of Frequency Hopping and Direct Sequence Spread Spectrum Modulation for IEEE 802.11 Applications at 2.4 Ghz", Harris Semiconductor Palm Bay, Florida.

[2]. K.C. Chen, "Medium Access Control of Wireless LANs for Mobile Computing", IEEE Network, September/October 1994.

[3]. W. Diepstraten, "A wireless MAC protocol comparison." -", Doc. IEEE P802.11-92/51, May 1992.

[4]. http://grouper.ieee.org/groups/802/index.html.

[5]. IEEE Standards Department, IEEE Std. 802.11-1997, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications".

[6]. IEEE Standards Department, IEEE Std 802-1990, Local and Metropolitan Area Networks: IEEE Standard Overview and Architecture.

[7]. L. Kleinrock, F.A. Tobagi, "Packet Switching in Radio Channels: Part I - Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics", IEEE Trans. On Communications, Vol. COM-23, No. 12, December 1975. L

[8]. R.L.Pickholtz, D.L. Schilling, L.B. Milstein, "Theory of Spread-Spectrum Communications - A Tutorial", IEEE Trans. On Communications, Vol COM-30, No.5, May 1982.

[9]. C.A. Rypinski, "Comments On - A Short Tutorial On CSMA -", Doc. IEEE P802.11/91-56, May 1991.

[10]. D.L. Schilling, L.B. Milstein, R.L. Pickholtz, M. Kullback, F. Miller, "Spread Spectrum for Commercial Communications", IEEE Communications Magazine, April 1991.

[11]. R.Scholtz, "The Origins of Spread-Spectrum Communications", IEEE Trans. On Communications, Vol. COM-30, No.5, May 1982.

[12]. J. Weinmiller, H. Woesner, J.P. Ebert, A. Wolisz, "Analyzing the RTS/CTS Mechanism in the DFWMAC Media Access Protocol for Wireless LANs", Electrical Engineering Department, Technical University Berlin.

[13]. B. Venners. Inside the Java Virtual Machine, McGraw Hill, New York, NY, 1998.

[14]. K. Arnold,J. Gosling. The Java Programming language, Addison-Wesley, 1997

[15]. T.Lindholm, F.Yellin. The Java virtual Machine specification, Addison-Wesley, 1997