An integrated development environment for the NS-2 *Network Simulator*

D. M. Oliveira; R. S. Cruz; R. J. P. B. Salgueiro; T. Rocha

Departamento de Computação, Universidade Federal de Sergipe, 49100-000, São Cristóvão-Se, Brasil

danilomo_@hotmail.com

ruironaldi@gmail.com

ricardo.salgueiro@gmail.com

tarcisio@dcomp.ufs.br

Simulation is one of the most important tools used for modeling and performance evaluation of computer systems. Among the various options for languages and simulation packages available for the area of computer networks, the NS-2 stands out for being the most popular open source tool used. However, the NS-2 does not provide a good infrastructure for the development of simulation projects. This paper proposes an integrated development environment (IDE) with a rich graphical interface that support project creation, distributed or sequential execution and result project recovery under the form of charts. The developed tool is used in a real project simulation performance evaluation of an access point for mobile devices via a Bluetooth connection. With this case study, the benefits introduced with the use of IDE in conjunction with the NS-2 are demonstrated.

Keywords: network simulation; NS-2; IDE

A simulação é uma das mais importantes ferramentas utilizadas para a modelagem e avaliação de desempenho dos sistemas informáticos. Entre as diversas opções de idiomas e pacotes de simulação disponíveis para a área de redes de computadores, o NS-2 se destaca por ser a ferramenta de código aberto mais popular usado. No entanto, o NS-2 não oferece uma boa infra-estrutura para o desenvolvimento de projetos de simulação. Este trabalho propõe um ambiente de desenvolvimento integrado (IDE) com uma rica interface gráfica que a criação de apoio ao projecto, execução distribuída ou seqüencial e recuperação de projeto resultado sob a forma de gráficos. A ferramenta desenvolvida é usada em uma verdadeira avaliação de desempenho do projeto de simulação de um ponto de acesso para dispositivos móveis através de uma conexão Bluetooth. Com este estudo caso, os benefícios introduzidos com o uso de IDE em conjunto com o NS-2 são demonstradas.

Palavras-chave: simulação de rede; NS-2; IDE

1. INTRODUCTION

Systems Information and Communication Technology (ICT) are becoming increasingly ubiquitous and present in people's lives and more and more people are depending on such systems. In these scenarios, users and system administrators are interested on an efficient and reliable operation. However, to ensure efficiency and reliability is not always an easy task, considering that the development of computer systems is not trivial. It is widely common that after a long cycle of analysis and implementation, the delivered systems do not operate with satisfactory levels of performance. Modeling systems, using models of performance, reliability, cost and availability, when applied to the early stages of development, helps to ensure that systems are delivered to the users in accordance to non-functional requirements [8]. These models also help to forecast the performance of the system at a future stage of greater use, providing guidelines for the administrator to modify the system to ensure to users the same conditions of use.

Simulation is a technique that consists in emulating the dynamic behavior of a system through a computer program. In the context of modeling and performance evaluation, simulation models are useful to gain a deeper knowledge about the system, conducting experiments that would not be possible with the real system [7]. This knowledge can be useful to compare design alternatives or to choose the value of a parameter that influences the system so that it can show its best performance. Once it is built and validated, a simulation model can be helpful in answering questions like "What if ...", that is, checking the system behavior in response to changes in system policies defined in the model, or changes in their configuration parameters.

In the area of computer networks, NS-2 [9] became one of the most widely used network simulators [5]. Thanks to its modular architecture, the code of the NS-2 can be extended with modules developed by researchers. These modules can implement several protocol layers: physical, data link, network, transport and application. Thus, the NS-2 has become one of the best network simulation environments available. However, the NS-2 has only a command line tool - the simulator itself - and not offer any support for managing simulation projects [10, 11]. This implies some degree of difficulty and low productivity for the developer who uses this tool.

In this paper we propose an integrated development environment (IDE - Integrated Development Environment) for the NS-2. This environment provides the following features: creation of projects based on simulation models and their parameters, integration with a persistence framework for storing and retrieving results, generation of reports and graphs about simulation results, and integration with a distributed execution environment for independent replication of simulations.

This paper is organized as follows. Section 2 discusses the simulation of computer networks and the related work. Section 3 describes the proposed tool - the IDE for simulations using the NS-2. In section 4, we show the case study designed to demonstrate the increased productivity and functionality offered by the IDE. Finally, we present the concluding remarks of this paper and present future work.

2. SIMULATION OF COMPUTER NETWORKS

Once simulation is chosen as modeling technique, the next decision to be made is about the tools for the development of simulation models. There are three approaches to the development of simulations: general purpose languages, simulation languages and simulation packages [6]. The advantage of using a general purpose language like C or Java, for example, is the flexibility and speed of execution of the simulation. Another factor that may lead to the choice of a general purpose language is the programmer's familiarity with the language. The disadvantage is the lack of facilities for the simulation of systems, where everything must be implemented by the programmer. Simulation languages are designed for computer simulation an offer features such as scheduler, events and a simulation clock, in addition to providing facilities to collect statistics of the simulation. Simulation packages are created to meet a specific problem domain. For example, simulation packages for computer networks already have the implementation of models of the physical layer of the TCP / IP and application protocols, allowing the user to reuse these models and apply them to a particular network scenario.

Creating a simulation model of a computer network from scratch with a general purpose language or simulation is a task that demands much effort. Fortunately, there are several tools available that simulate the behavior of the TCP / IP and various emerging networking technologies such as sensor networks, mesh networking and mobile IP. Several simulators have been proposed, each with its own characteristics and advantages. The network simulator OPNET [12] is quite popular and feature-rich, but it is a commercial tool. The NS-2 [9] represents the open source and freely distributed network simulator quite popular among researchers. Despite it is a relevant network simulator, the NS-2 do not offer support for managing simulation projects. An alternative to NS-2 and OPNET is Omnet + + [13]. The Omnet + + has free license for noncommercial purposes. This tool presents a greater support to the development of simulation that the NS-2, which concentrates its efforts in network simulator and simulation models, but is not concerned with the issue of supporting the developer of simulations. Considering the popularity and relevance of the NS-2 among researchers, the objective of this work is to provide an infrastructure for the NS-2, with the following features:

organization of simulations, distributed execution, collecting results and generating reports and graphics, embedded in an IDE.

The Network Simulator (version 2), better known as NS-2, is a network simulator based on events that was built from the REAL network simulator in 1989. Since then, several researchers and institutions such as the Defense Advanced Research Projects Agency (DARPA) and National Science Foundation (NFS) contributed to its development. The NS-2 provides mechanisms for simulations of wired and wireless networks, applications, traffic patterns, routing algorithms, multicast protocols, etc. It is possible to model and estimate the performance of computer networks through simulations and to validate protocols.

The NS-2 can be used as follows. The user creates a simulation script using the language OTcl, where he declares the topology of the model and the network applications that run on this topology and performs the necessary settings. He then gives that script to the simulator. The script is processed by the OTcl interpreter, which instantiates the event scheduler and other objects and run the simulation. At the end of the simulation, the descriptions of all the events in the simulation are recorded in files. These files can be text-based or based on animation. The first type is useful when you want to create graphics or perform statistical calculations on the simulation results. The second type is given as input to the tool NAM (Network Animator), which generates an animation of the simulation. Figure 1 describes the operation of the NS-2.

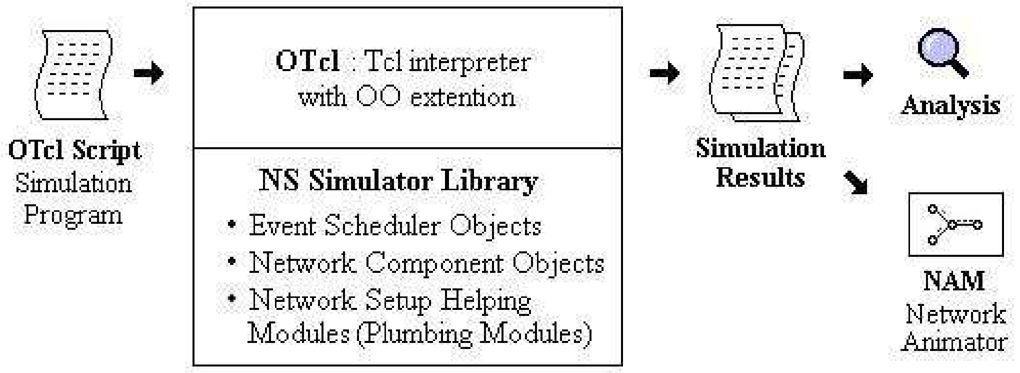

OTcl Script
Simulation Program
OTcl : Tcl interpreter with OO extention
NS Simulator Library
• Event Scheduler Objects
• Network Component Objects
• Network Setup Helping Modules (Plumbing Modules)
Simulation Results
Analysis
NAM
Network Animator

Figure 1: Overview of the functioning of the NS [4]

In a study of modeling and performance analysis, using the NS-2 differs from the scenario described above. In this case, a model represented by a simulation script is created, and some configuration parameters and the workload offered to the system are pending. A parameter is associated with a list of values it can take. For each combination of parameter values, you can run the simulation scenario and get the performance metrics to be investigated. These metrics can be represented by the flow of input/output data in an application or network link, packet loss rate, the delay that a packet suffers from the source node to destination node, etc.. In order to ensure the accuracy of the simulation results, each individual simulation can be replicated several times, keeping the same parameter values and using a random numbers generator.

Despite NS-2 is the most popular open source network simulator, it does not offer any tool that facilitates the creation of simulation projects and extraction of its results in an automated fashion. These tasks are delegated to the developer and they are usually performed through scripts in bash, Perl or Python, which is are troublesome and error-prone. To overcome this difficulty, we developed the NS-Facilities tool [10] - a framework for creating simulation projects in NS-2. With this tool, we use a script of the NS-2 simulation with open parameters, and a configuration file that defines the list of parameters and their values. For each combination of parameters and their replication, the framework creates an instance of the stage and performs the automated execution of all possible scenarios. The framework also makes the extraction of the results and stores them in a relational database, facilitating later retrieval of data. Figure 2 illustrates the operation of the NS-Facilities.

Another problem involved in using the NS-2 is the fact that the sequential execution of independent replications can result a high computational cost, especially in more complex projects. Unlike NS-2, independent replications could be performed on different machines, reducing the overall time for project execution. This approach is known in literature as MRIP - Multiple Replications in Parallel [3]. The tool NS-DiS [11] is an extension of the tool NS-Facilities that follows this approach. In Figure 3 we show the topology of the NS-DiS. Its operation is described below.

- The user submits a project for distributed execution through the client module of the application;
- The server accepts this submission and performs processing. It divides the project into equal parts, based on the number of client nodes registered in the environment;
- The server distributes the shares among the slave nodes, which are machines that offer the processing power to the server simulation;
- The slave nodes accept submissions from the simulation server, perform the execution of the replication received, process and return its results to the server;
- Finally, the server stores the collected metrics in a relational database that can be available for users to query later.

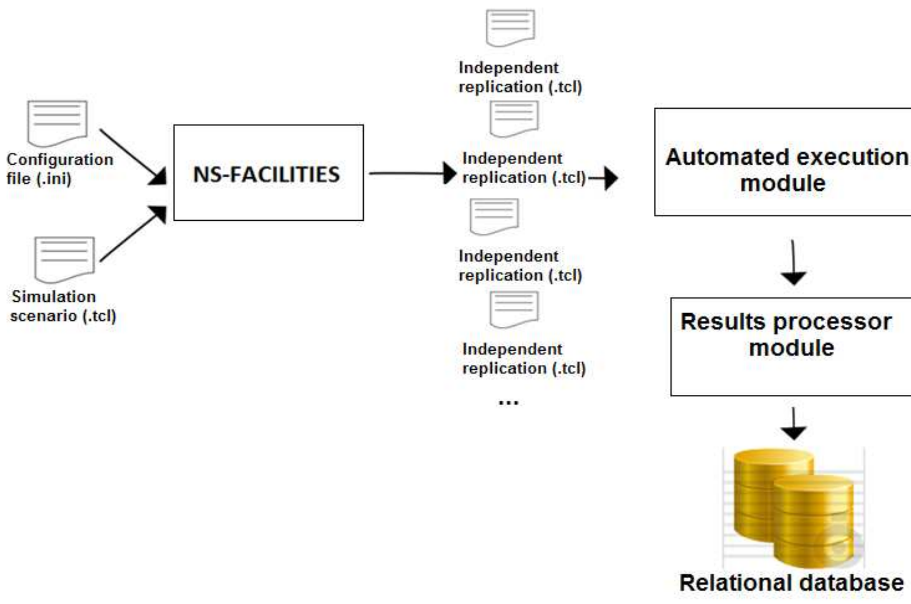

Configuration file (.ini)
Simulation scenario (.tcl)
NS-FACILITIES
Independent replication (.tcl)
Independent replication (.tcl)
Independent replication (.tcl)
Independent replication (.tcl)
...
Automated execution module
Results processor module
Relational database

Figure 2: Operation of NS-Facilities

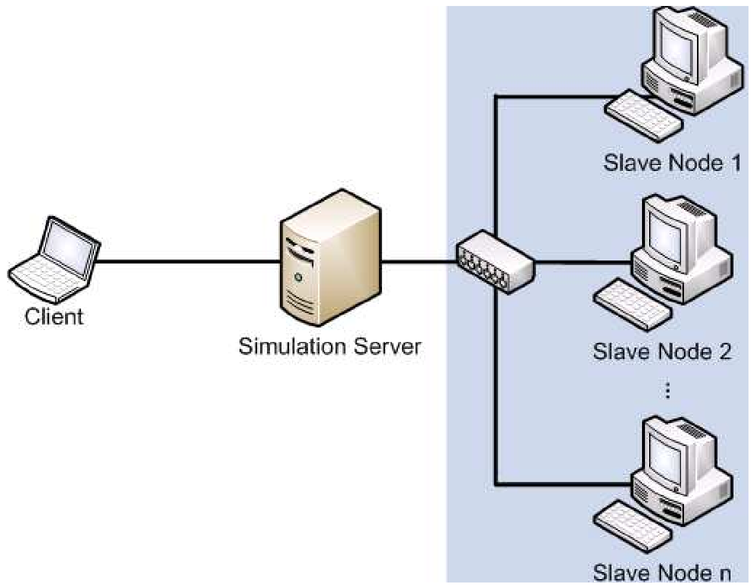


Figure 3: Topology of the distributed simulation environment NS-DiS[11]

In order to provide easier to use the NS-DiS, we create an integrated development environment that has a rich user interface, allowing more productive use of the NS-2 and NS-DiS. This environment is detailed in the following section.

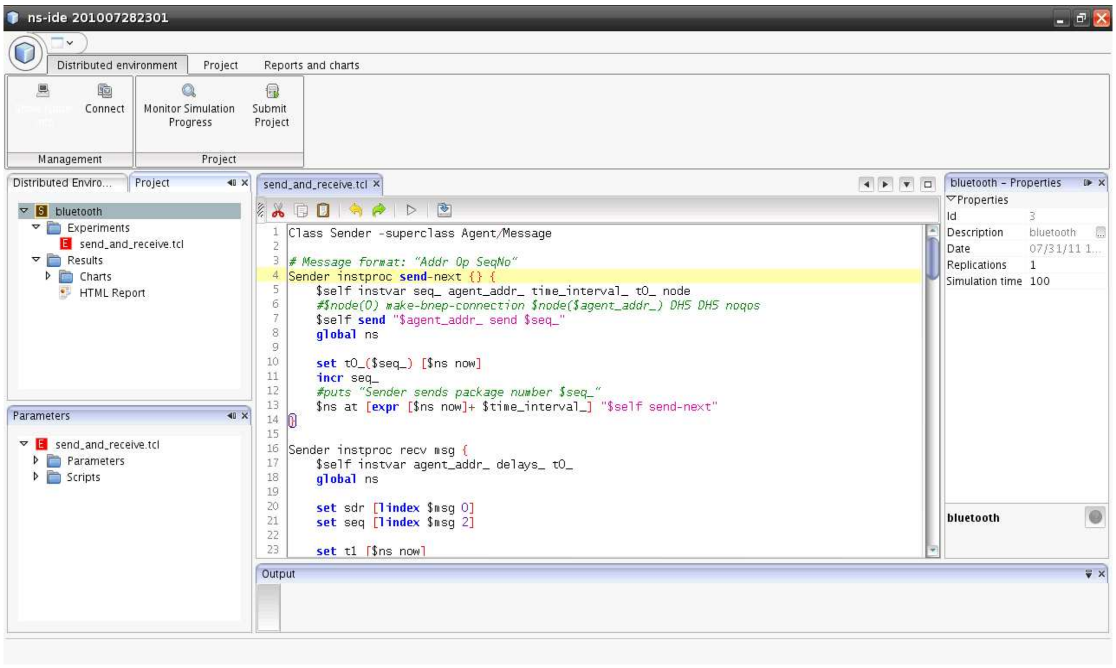
3. INTEGRATED DEVELOPMENT ENVIRONMENT FOR SIMULATION

An IDE (Integrated Development Environment) is a tool that incorporates many features required to build software in a particular language. The main features offered by an IDE are the program editor, execution environment, debugging environment, project management, among others that vary from IDE to IDE. To facilitate the use of NS-2 and allow the development of more complex simulation projects, we developed an integrated development environment for NS-2. Our IDE is built on the Netbeans platform [2] which is a RCP (Rich Client Platform) to build more sophisticated desktop applications. This platform defines a modular architecture for the application and allows the reuse of components of the Netbeans IDE (which was built on the Netbeans platform).

In Figure 4 we show the main screen of the IDE. It is composed by a toolbar at the top of the application, an editor at the center and small docked windows around it. The toolbar of the IDE is divided into categories related to the various features of the IDE. These categories are described below.

- ***Project*** **-** This category contains commands related to creating, configuring and running local simulation projects. A project open or newly created will be shown in the “Project” box (Figure 5 (a)). Each project consists of one or more models, which consist of a parameterized simulation scenario. The scenario is described by a simulation script in NS-2 OTcl language and can be edited in the central window of the IDE. The parameters of the scenario are displayed in the “Parameters” (Figure 5 (b)) box. The values that the parameter can take are set in the “Properties” box (Figure 5 (c)).

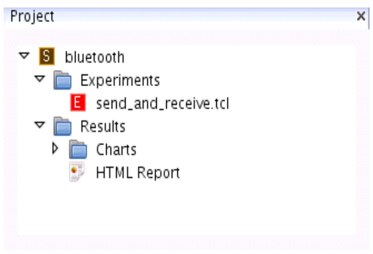
- ***Distributed Environment*** - In the box “Distributed Environment” (Figure 6 (a)), commands related to the distributed simulation environment are located. The button “Connect” can be used to connect or disconnect the nodes of the environment. The environment is shown in the side window in Figure 6 (b). For each node that symbolizes a slave node, there is an option to view the current situation of the node, if it is busy or not and how many replications it keeps in its queue. There is also the option to terminate the slave node program, or even suspend the entire environment, ending the simulation server and all its slave nodes at once. The other two buttons on the toolbar allows you to submit the project to the remote environment and to monitor its running.



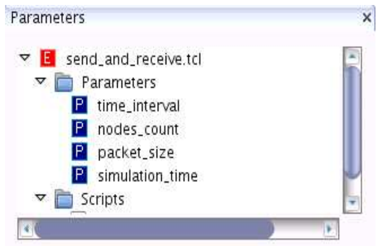
```
Class Sender -superclass Agent/Message

# Message format: "Addr Op SeqNo"
Sender instproc send-next {} {
    $self instvar seq_ agent_addr_ time_interval_ t0_ node
    #$node(0) make-bnep-connection $node($agent_addr_) DH5 DH5 noqos
    $self send "$agent_addr_ send $seq_"
    global ns

    set t0_($seq_) [$ns now]
    incr seq_
    #puts "Sender sends package number $seq_"
    $ns at [expr [$ns now]+ $time_interval_] "$self send-next"
}

Sender instproc recv msg {
    $self instvar agent_addr_ delays_ t0_
    global ns

    set sdr [lindex $msg 0]
    set seq [lindex $msg 2]

    set t1 [$ns now]
```

Id	3
Description	bluetooth
Date	07/31/11 1...
Replications	1
Simulation time	100

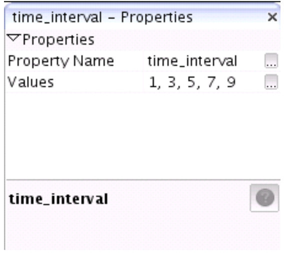
Figure 4: Screenshot of simulation IDE



(a) Project explorer



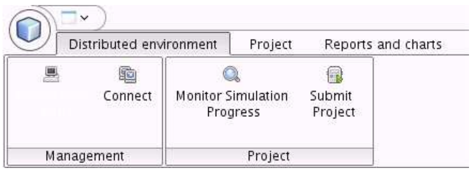
(b) Parameters inspector



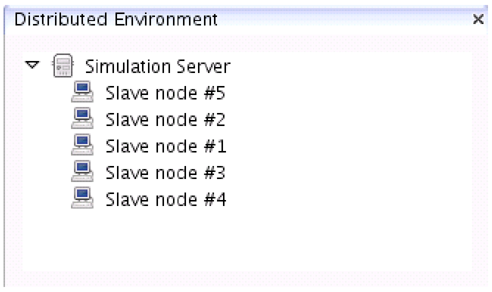
Property Name	time_interval
Values	1, 3, 5, 7, 9

(c) Properties

Figure 5: Project explorer windows



(a) Distributed environment Toolbar



(b) Distributed environment window

Figure 6: Distributed environment toolbar and window

- ***Charts*** - Simulation models are useful to help managers and system administrators to make decisions about system design and resource policies. In order to support the decision making process, the results should be presented clearly and understandably. The results are presented under the form of charts and tables. This is an additional stage in the project that requires practice with the graphic tools and word processing, which can be annoying and repetitive. Therefore, our IDE offers a module to automate the generation of graphics. The functioning of this module is described in Figure 7. The module takes the formula of the graph as input such as its curves, labels and variables, and the input data that are the results of the simulations. It generates a as a result a chart with this data, in accordance to the formula specifications. In Figure 8, the main step of graphing functionality is displayed. Figures 10 and 11show examples of generated graphics with this module.

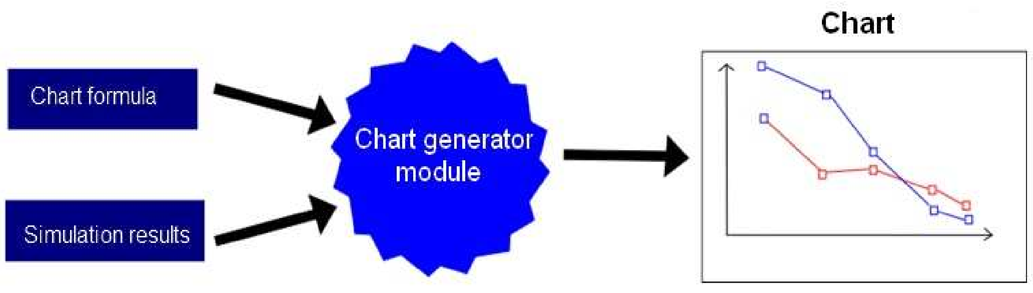
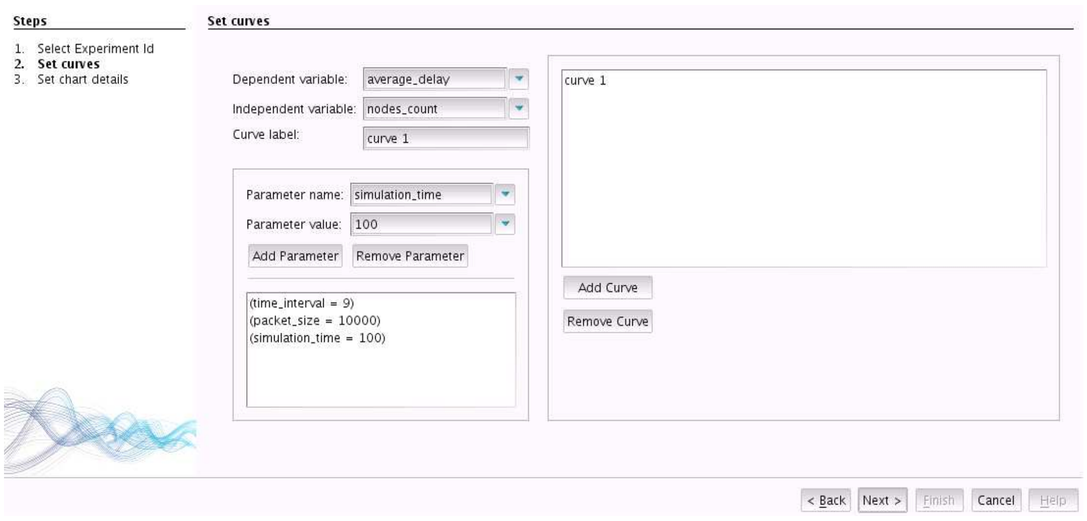


Figure 7: Chart generator module behavior



Dependent variable: average_delay
Independent variable: nodes_count
Curve label: curve 1
Parameter name: simulation_time
Parameter value: 100

```
(time_interval = 9)
(packet_size = 10000)
(simulation_time = 100)
```

curve 1

Figura 8: Chart generator wizard

4. CASE STUDY

The case study consists of a real NS-2 application built using the IDE and its features, demonstrating its effectiveness in increasing productivity. Bluetooth network was the test scenario. Widespread in mobile phones, they can offer Internet connectivity at a reduced cost, as described in [1]. The objective of this case study is to investigate the feasibility of this scenario, and examine the maximum data throughput that a Bluetooth proxy server is able to offer its customers while guaranteeing a acceptable delay.

The experiment consists of a scenario with a server node and one or more client nodes. Each client sends requests to the server at certain intervals of time, through a Bluetooth connection. In this request it specifies the number of bytes it wishes to receive in the reply message. This process is illustrated in Figure 9.

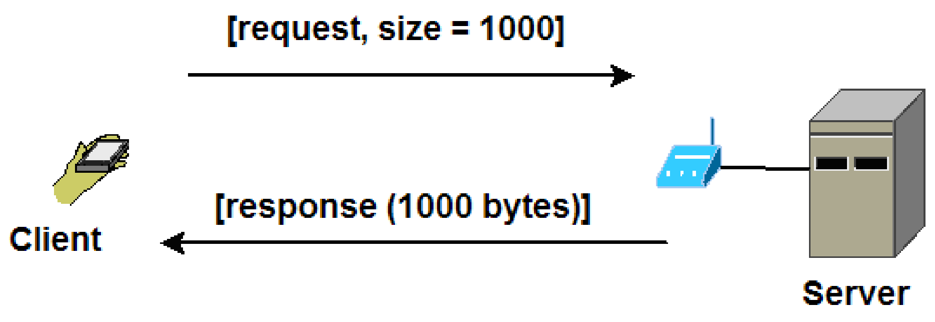


Figure 9: Client server interaction over a bluetooth connection

The simulated scenario has three parameters:

- ***time interval*** - Time interval between requisitions;
- ***nodes count*** - Number of client nodes. This can be at most 7, because 8 nodes (1 master, 7 slaves) is the maximum that a Bluetooth piconet can support;
- ***packet size*** – The response message length.

Table 1 shows the levels on which each parameter varies in the experiments.

Table 1: Parameters and levels

Parameter	***Levels***
Time_interval	1, 3, 5, 7, 9 (seconds)
Node_count	1, 2, 3, 4, 5, 6, 7
Packet_size	10000, 20000, 30000, 40000, 50000 (bytes)

A full factorial design was constructed. Each element of the Cartesian product of parameter levels generates a measurement, and each measurement was repeated five times. For each replication the following performance results were collected:

- ***Average delay*** - The average delays of all packets sent. Each delay is measured by calculating the difference between the time the request message is sent and the time the response is received;
- ***Out throughput*** – The throughput in the server.

4.1. Results

To investigate the impact of throughput on the server on the value of average delay on the clients, two charts were created. The first (Figure 10) is a graph that shows the throughput depending on the size of the response message returned by the server, into a scenario with seven active clients. We measured out throughput for each value of time interval parameter and displayed as curves in the chart. While decreasing time interval and increasing the size of the reply message we increase the out throughput. Our goal is to find a value for the flow that implies an acceptable delay on the client. In Figure 11 we display a chart that presents the same variable (size of the response message) based on the average delay, also into a scenario with seven client nodes. The curve which shows the measurements with the interval between requests defined as 1 second displays very large delays, sometimes exceeding the value of 40 seconds. We observed that the delay reaches very high values for a 3 seconds time interval and the message size response is greater than 30,000 bytes.

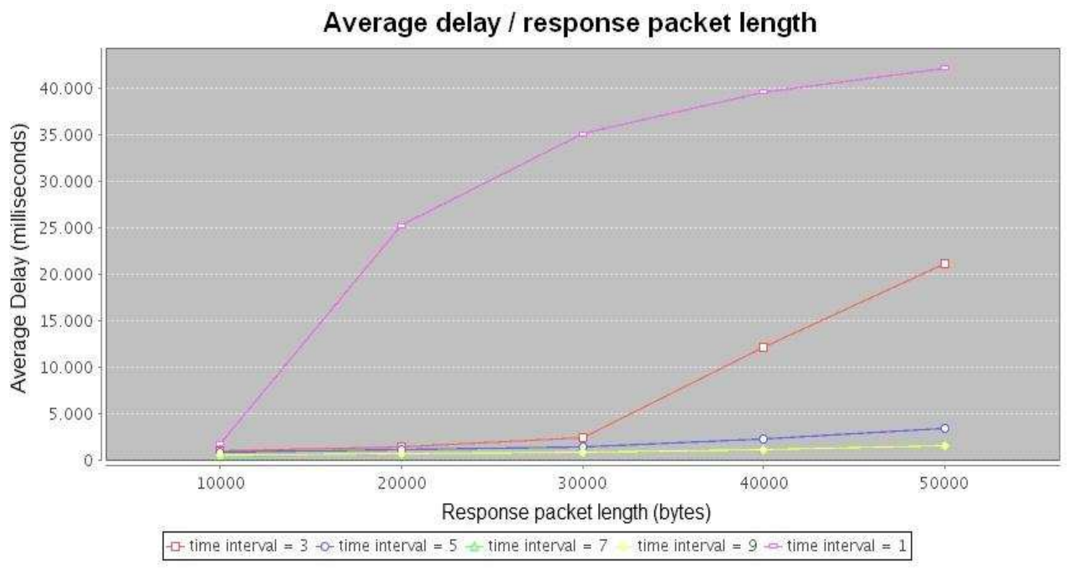


Figure 10: Out throughput vs. Response packet length

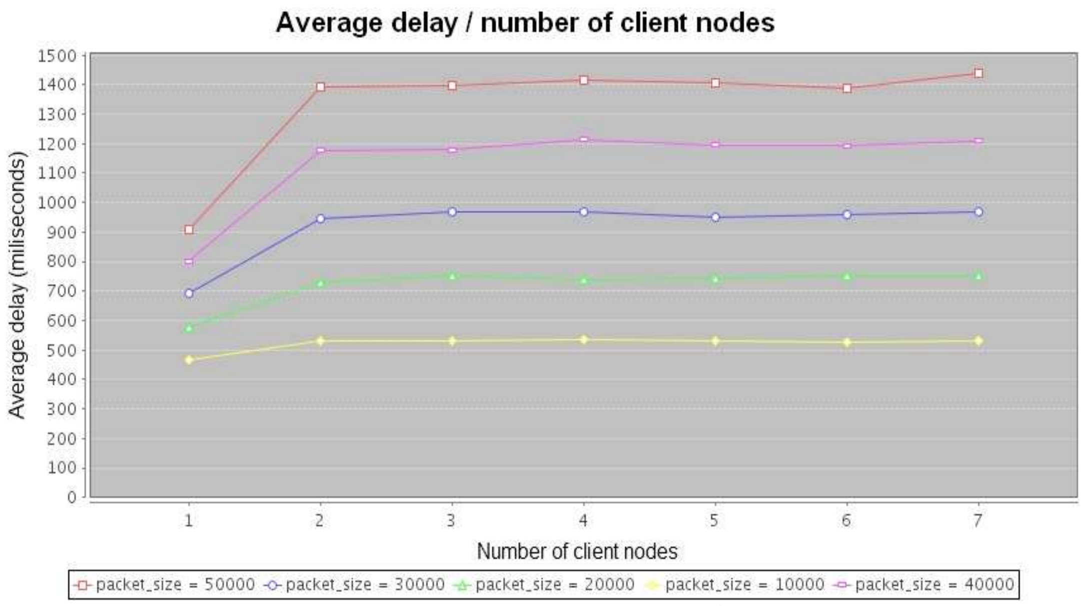


Figure 11: Average delay vs. number of clients

Assuming we determine the maximum acceptable delay value to 2 seconds we can determine the maximum throughput with the SQL query listed in Figure 12. This query returned a value of 60,422 B/s, which means that if the out throughput on the server exceeds this limit, the delay will surpass the time of 2 seconds.

```
select max(value_scalar) from SCALAR
where DSC_SCALAR = 'vazao_saida'
and id_study = 3
and id_measurement in (select id_measurement from scalar
where id_study = 3
and dsc_scalar = 'average_delay'
and value_scalar < 2000)
```

Figure 12: Query of the higher throughput from which average delay is less than 2 sec.

5. CONCLUSIONS

System simulation is an important tool for performance evaluation in computer networks. Among the simulation tools available for this purpose, the NS-2 stands out by its modular architecture and the wide range of simulation models available, making it the network simulator chosen by several researchers in their projects. However, the NS-2 still has limitations on the infrastructure to support the needs of simulation developers. We have seen that a simulation study not only uses a single simulation scenario, it's generally based on an experimental project that defines several specific instances of simulation. As the NS-2 provides no such support, the developer needs to develop a series of scripts for organizing the experiments, performing the automated execution, carrying out the processing of trace files generated, summarizing this data, and generating charts.

This paper presents a versatile integrated development environment for the NS-2 which aims to overcome the limitations described above. The IDE has the following features: a project manager that allows you to create simulation models and describe their parameters, local execution of the project, executing the project in a distributed environment, monitoring of distributed environment, and recovery of the simulation results through queries and charts. With the case study, we found that the use of IDE in conjunction with the NS-2 and NS-DiS allows the project developer to concentrate on the details of the modeled system and the experiments, avoiding the trouble of creating routines to perform the tasks related to configuration parameters, automatic execution of the replication, storage and graphing results. In conjunction with the distributed environment, the IDE provides greater productivity and allows more complex problems are developed.

---

1. AULETTA, V.; BLUNDO, C.; CRISTOFARO, E. D. Http over bluetooth: a j2me experience. In: IARIA. The International Journal On Advances in Telecommunications. [S.l.: s.n.], 2008.
2. BOUDREAU, T.; TULACH, J.; WIELENGA, G. Rich client programming: plugging into the netbeans platform. Prentice Hall Press Upper Saddle River, NJ, USA, 2007.
3. BRUSCHI, S. M.; SANTANA, R. H. C.; SANTANA, M. J.; AIZA, T. S. An automatic distributed simulation environment. In Proceedings of the 36th conference on Winter simulation (WSC '04). Winter Simulation Conference, 2004
4. CHUNG J.; CLAYPOOL, M. Ns by example. Online tutorial. December 2008.
5. ISSARIYAKUL, T.; HOSSAIN, E. Introduction to Network Simulator NS2. [S.l.]: Springer,2009. ISBN 978-0-387-71759-3.
6. JAIN, R. The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling. [S.l.]: Wiley New York, 1991.
7. LAW, A.; KELTON, W. Simulation modeling and analysis. [S.l.]: McGraw-Hill New York, 1991.
8. MENASCE, D. et al. Performance by design: computer capacity planning by example. [S.l.]: Prentice Hall, 2004
9. NS-2. Official website of the project. Available at [http://nsnam.isi.edu/nsnam]. 2008.
10. OLIVEIRA, D.; CRUZ, R.; SALGUEIRO, R. Ns-facilities - uma ferramenta de apoio ao desenvolvimento de simulações. In: ESCOLA REGIONAL DE COMPUTAÇÃO BAHIA ALAGOAS SERGIPE. WTICG 2010. [S.l.: s.n.], 2010.
11. OLIVEIRA, D.; SALGUEIRO, R.; ROCHA, T. Ns-DiS: um ambiente de simulação distribuído para o ns-2. In: FÓRUM INTERNACIONAL DE SOFTWARE LIVRE. Workshop de Software Livre. [S.l.: s.n.], 2011.
12. OPNET MODELER. Official website of the project. Available at [http://www.opnet.com], 2009.
13. VARGA, A. et al. The OMNeT++ discrete event simulation system. In: Proceedings of the European Simulation Multiconference (ESM 2001). [S.l.: s.n.], 2001. p. 319–324.