Network Simulation By Mouse (NSBM): A GUI Approach for Teaching Computer Networks with the *ns* Simulator

Michael Welzl, Muhammad Ali, Sven Hessler
Institute of Computer Science
University of Innsbruck, Austria

Abstract

The ns network simulator is a common tool frequently used for teaching the fundamentals of computer networks. However, our experience with students has shown that writing long ns scripts is not appreciated by them. Hence, we used a custom-made visualization tool — NSBM — where topologies are created using a mouse and then the TCL script is generated automatically. The tool is interactive and has an easy interface to create topologies and define agents and protocols that govern the communication setup. The feedback of students was relatively encouraging as the majority found it to be useful in learning networking fundamentals.

1 Introduction

The *ns-2*[1] network simulator has been used by network researchers for a long time for implementing and analyzing novel ideas and new approaches emerging in computer networks. Likewise, the tool is widely used in universities for teaching networks. A course like computer networks, can become extremely boring and, sometimes, frustrating when there is no visualization of the contents that the students study in their books. Moreover, using simulation tools like *ns-2* makes it

[1] sometimes called *ns* and sometimes *ns-2* referring to the version number; we will use the term synonymously

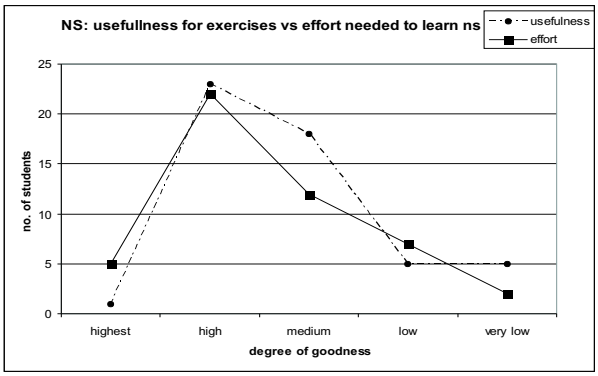

Figure 1: **Students feedback comparison of effort required to learn *ns* against its usefulness**

easier to make students understand the theoretical aspects that they learn. Our experience of teaching computer networks using *ns-2* along with other similar tools like CAVT [1] and IRVtool[2] has further acknowledged its significance.

The convenience of using *ns-2* in teaching, nevertheless, comes with a price. We have learned through our personal teaching experience of several years that *ns-2* has never been a favorite choice for students. They disliked it because of the lengthy coding of TCL language. Students feedback clearly indicates their discomfort over the huge amount of effort required to learn *ns-2* despite the fact that they still consider it a very useful tool for solving their exercises as shown in figure 1.

Keeping in view the above-stated fact, we decided to build a tool that is simpler and more helpful to work with — the solution was *NSBM (Network Simulation By Mouse)*.

2 Using *ns* for teaching

ns[3] is an object-oriented, discrete event driven network simulator developed at UC Berkely and written in C++ and OTcl. *ns* is a very common tool used for simulating local and wide area networks. It implements network protocols such as TCP and UPD; traffic source behavior such as FTP, Telnet, Web, CBR and VBR; router queue management mechanism such as Drop Tail, RED and CBQ; routing

[2]**I**nternet **R**outing **V**isualization tool available on http://www.welzl.at/research/tools/irvtool

[3]http://www.isi.edu/nsnam/ns

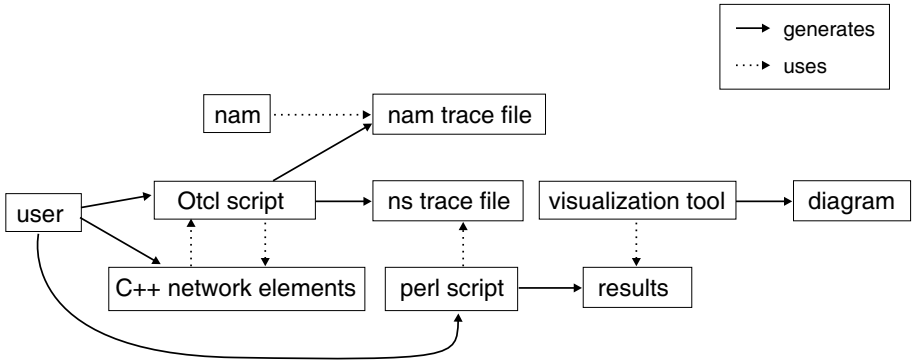

Figure 2: **A typical *ns* usage scenario**

algorithms such as Dijkstra, and a lot more. *ns* also implements multicasting and some of the MAC layer protocols for LAN simulations. The simulator is open source, hence, allowing anyone and everyone to make changes to the existing code, besides adding new protocols and functionalities to it. This makes it very popular among the networking community which can easily evaluate the functionality of their new proposed and novel designs for network research. The simulator is developed in two languages: *C++ and OTcl*[4]. C++ is used for detailed implementations of protocols like TCP or any customized ones. TCL scripting, on the other hand, is the frontend interpreter for *ns* used for constructing commands and configuration interfaces. For example, if you want to develop a new routing protocol, you have to write it in C++ and add it into the *ns* library. In order to check the functionality of this protocol, you use TCL scripting through which you can create the required topology, define parameters for links and nodes, and perform simulations to realize your own protocol in action. Furthermore, a *Network AniMator* (NAM) is also provided with *ns* in order to visualize and interact with the system at run-time. Finally, graphs can be created from the produced results to evaluate and analyze the performance of the system by using *xgraph*. A typical simulation process is shown in figure 2.

Besides its numerous advantages for network researchers, working with *ns* is a difficult job mainly due to lack of good documentation at the beginner level. The user manual available at the *ns* main website is too difficult to comprehend by students who are only interested to gain some hands-on experience to practice what they learn in theory. We tried to alleviate this problem by providing our

[4]Tcl script language with object-oriented extensions developed at MIT

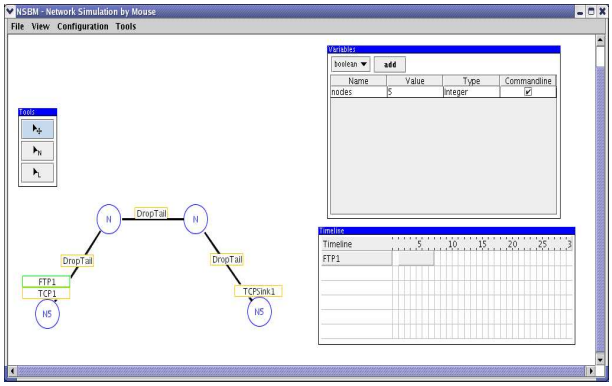

Figure 3: **NSBM Main-screen**

own documentation in German language,[5] but that did not help much. This may be attributed to the fact that the documentation is not the only problem – the procedure for carrying out simulations is just exertive, and this may be an issue for students. Having said that, our experience in the class clearly shows that *ns* is an inappropriate tool for teaching exercises due to its cumbersome usage and non interactive nature. The fact is further extended by the feedback we collected from the students. This brought us to realize a handy and interactive tool which is easy to teach and use in the lectures. We call this tool as NSBM — *Network Simulation By Mouse*

3 NSBM

NSBM, developed in java, is a graphical tool that is used to generate TCL script using a mouse. Nodes and links can be created with a single mouse click. When the program is started, a screen appears with all the tools available on it as shown in figure 3.

- Display area: This is the area available to the user to create nodes, links and define protocols and agents governing the communication using a mouse. Right-clicking on an element displays its properties, whereas with a left click of mouse, an element is created.
- Tools: There are three mouse pointer options available in the tools menu; if you select the first one, you can move the node freely on the display area

[5] http://www.welzl.at/congestion

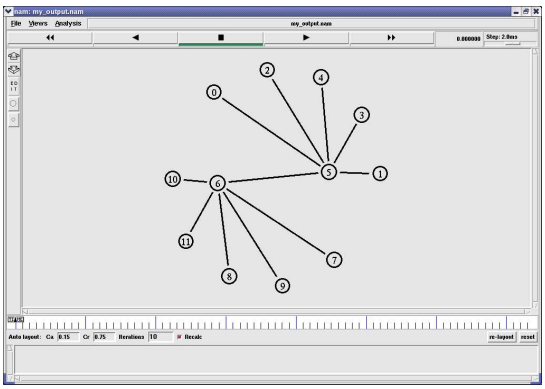

Figure 4: **A screenshot of *nam***

and the link will accordingly follow. The second option lets you create a new node and the third one lets you create a link between nodes.

- Variable: These are used to define global variables for the simulation environment. NSBM lets the user to define how many nodes a ”node” represents, for instance, in figure 3, the node labeled ”N5” represents 5 nodes. This number can depend on a global variable, which can also be a command line argument for the *ns-2* TCL script.

- Timeline: The timeline window shows all the ”applications” and their time slices. In figure 3, for instance, the application FTP1 starts at one second and ends at seven seconds. Also new ”run events” can be created by using right mouse button.

4 NSBM usage

Consider Figure 3 which represents a simple dumbbell topology with N FTP senders and receivers. After creating the topology and defininig all the agents/ protocols in the canvas, ‘Generate NS Code’ is executed from the ‘File’ menu which automatically creates the ‘code.tcl’ file. After this, *ns* is executed with the following command:

```
ns code.tcl number_of_nodes
```

Figure 5: **A screenshot of *xgraph***

where 'number_of_nodes' represents the number of source and destination nodes.

In the current scenario, the following commands were executed:

```
ns  code.tcl   5
nam  out.nam
```

where 'code.tcl', the generated TCL file is called with five nodes each for the sender and receiver. The name of the *nam* output file was set to 'out.nam'.

Also, with 'out.tr' being the name of the regular *ns* trace file specified in NSBM, typing

```
perl  ./throughput.pl     out.tr   tcp  6 > graph
xgraph   graph
```

gave Figure 5, which is a screenshot of *xgraph*. As shown in Figure 4, node number 6 is one end of the bottleneck; pressing the 'Play' button in *nam* clearly shows that it is the receiving end, as one can see packets flying from nodes 0 to 4 to nodes 7 to 11.

We have seen that NSBM and the 'throughput.pl' script make it very easy for students to create *ns* scenarios, the results of which can be plotted with *xgraph* and

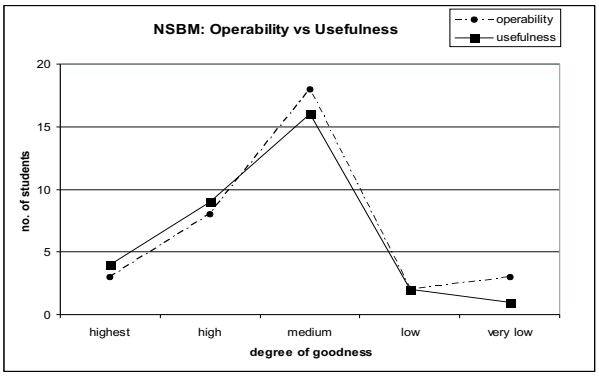

Figure 6: **Students feedback comparison of operability of NSBM against its general usefulness**

interactively studied with *nam*. All in all, it seems that the *ns* experience has turned from 80% batch processing (not 100% because of *nam*), to almost fully interactive usage. Moreover, it seems that using *ns* has become quite straightforward.

5 Students reaction to NSBM

After providing an opportunity to our graduate students to use NSBM for solving their exercises, we collected their feedback regarding the tool. The students' reaction to NSBM was quite encouraging. An analysis for feedback collected from around 60 students shows that more than 50% of the students ranked NSBM between highest and medium in terms of its operability and usefulness as compared to only 10% who ranked it as "low" or "very low" as shown in figure 6. The result confirms our belief that teaching with visualization tools like NSBM is convenient and helpful for both teachers and students to learn the contents in an efficient way. This fact is further strengthened by our positive teaching experience with other visual tools, like IRVtool and CAVT. We therefore intend to extend this approach to encompass all the components of networking into a single completely interactive simulator.

6 Conclusion

The design of NSBM was guided by the goal of making its usage as easy as possible. There are some other GUIs for network simulation available (e.g. the nam editor [2], nsbench [3] and omnet++ [4]), but these tools were not fit for our purposes, as they either lacked some key functions or had too many features — forcing students to use a GUI that is hard to handle was absolutely not our intention.

Teaching is a learning process in itself; like any approach to enhance the experience for students, NSBM is merely a part of that process — a step in the right direction, but not the end of the journey. This fact is underlined by the feedback from students, which we consider as encouraging, but not good enough to leave NSBM unchanged. As the next step, we plan to change NSBM into a fully interactive simulation tool with basic embedded simulation capabilities; with this new tool, students would not even have to handle TCL files anymore, and no external simulator would have to be called — the simulation could be executed with a mouseclick, and the rest of the tool would still remain as simple as it currently is.

References

[1] Michael Welzl, Max Mühlhäuser: ”CAVT - A Congestion Avoidance Visualization Tool”, *ACM Computer Communication Review*, Volume 3, Issue 3, (July 2003), pp. 95-101.

[2] R. Canonico, D. Emma, G. Ventre, “Extended NAM: An ns2 Compatible Network Topology Editor for Simulation of Web Caching Systems on Large Network Topologies”, *European Simulation and Modelling Conference*, Naples, Italy, Oct. 2003.

[3] www.mnlab.cs.depaul.edu/projects/nsbench/

[4] www.omnetpp.org

Authors: Michael Welzl, Muhammad Ali, Sven Hessler
(Michael.Welzl,Muhammad.Ali,Sven.Hessler)@uibk.ac.at
Institute of Computer Science, University of Innsbruck,
Techniker Strasse 21-A, 6020, Innsbruck, AUSTRIA.