Simulation Design and Execution
The Case of TRAFIL

Christos Bouras[1,2], Savvas Charalambides[1,2], Michalis Drakoulelis[2], Georgios Kioumourtzis[3] and Kostas Stamos[1,2]

[1]*Computer Technology Institute and Press "Diophantus", D. Maritsas Building, N. Kazantzaki Str, University Campus, GR 26504, Patras, Greece*

[2]*Computer Engineering and Informatics Department, University of Patras GR 26504, Patras, Greece*

[3]*Center for Security Studies KEMEA, P.Kanellopoulou 4, GR 10177, Athens, Greece*

{bouras, stamos}@cti.gr, {charalampi, drakouleli}@ceid.upatras.gr, kioumourtzis.kemea@gmail.com

Keywords: Simulation Design, Analysis, Execution, Trace File, Ns-2, TRAFIL.

Abstract: This paper presents the enhancements that we have done on the TRAFIL (TRAce FILe) framework, which significantly extend its capabilities in the areas of simulation design and execution. In particular, TRAFIL now offers the possibility to design, create, execute and review NS-2 simulation scenarios, on top of its existing post-simulation trace analysis functionalities. The enhancements make TRAFIL a complete wrapper around the NS-2 simulator, allowing the user to perform all steps from pre-simulation design to actual simulation execution in an automated way and fast and convenient post-simulation analysis of potentially large amount of data. The paper describes the new TRAFIL architecture and how these enhancements were implemented with the goal of relieving NS-2 users from the often cumbersome tasks of script writing and validating, while also enabling them at to go behind the TRAFIL environment and into the simulator internals at any time. The paper presents the new GUI functionalities developed for that purpose, the approach that we took for their design and how these fit in the overall TRAFIL architecture.

1 INTRODUCTION

Network Simulation (Breslau, 2000, Fujimoto, 2003) constitutes a research approach where the behaviour of a real network is modelled using software. Often a simulation of the network's functionality is represented by calculating the interaction of its pieces and then analysing the performance.

Network simulation tools (Weingärtner, 2009, Azizur Rahman, 2009, Wang, 2007) are widely used for simulating physical networks with great ease. A very popular network simulation tool is NS-2 (NS-2 website, 2013, Cicconetti, 2006, Chen, 2007) which is a discrete event simulator used for research in computer networks. It can be used for simulating a variety of protocols such as multicast, Transmission Control Protocol (TCP) and routing in different kinds of networks. In order for a user to design a simulation for it to be executed via NS-2 the OTcl scripting language is used. With OTcl a user can define the various scenario components, the interaction of which comprises the actual simulation. A variety of components exists and typical network modules that are present in each simulation are Nodes, Links, Queues, Traffic Generators, Agents etc. These objects are user-defined and for many of them there is a variety of types from which a user can select to utilize in his/her scenario.

Nevertheless, a user has to learn how to write an OTcl script firstly and then identify how to use each component. Even for a small simulation a user has to understand NS-2 notation and the meaning for each object. This is discouraging for average users or for users who their main strength is not programming.

However, a number of tools have been developed so far (NS-2 Contributed Code, 2013) to allow either a flexible collection of NS-2 simulation traces or to provide a Graphical User Interface (GUI) for the end user to simplify simulation set up. To our knowledge no single tool exists that enables both simulation results and post-simulation storage and processing in a unified package, which can hide the underlying complexities entirely.

Therefore, the main focus of this paper is to present the new functionalities offered by TRAFIL that enable more convenient development of NS-2

scripts and the flexible execution of network simulations (Luis Font, 2011). TRAFIL now gives the opportunity for users to define the network topology, the interaction between network components, traffic flow and all other possible parameters via a graphical user interface. The purpose of this framework is to alleviate the daunting task of having to understand the underlying implementation, the specific notation and eventually the development of an OTcl script, and also make automated execution easier. Therefore, with the new additions TRAFIL can now automate the process from designing to executing a simulation, and then also the post-processing of results, which has been its initial focus. As a matter of fact, the user can now perform all tasks of code writing, code debugging, testing code using NS-2 and result processing, and can skip directly from network designing to viewing test results without a need to use the command line (although any of the above steps can still be performed manually).

The remainder of the paper is organized as follows: In section 2 we describe the overview of TRAFIL including its architecture with description of each component individually and the various features it offers. Section 3 introduces the new functionality of describing the simulation plane along with all the features offered. In section 4 we present how the simulation plane description is used to eventually execute the simulation. Section 5 presents various examples of how this framework can be used and the results that can be extracted. Finally Section 6 concludes the paper and discusses our future plans made.

2 TRAFIL OVERVIEW

Figure 1 presents the TRAFIL architecture. TRAFIL is organized in a 3-layer architecture. The presentation layer is responsible for handling the user interaction. All the user requests are made via the presentation layer and the results of each request are visible again through the presentation layer.

The presentation layer hands all the user requests to the business layer where all the processing takes place. TRAFIL offers a variety of functionalities and therefore the business layer is comprised of many different sub modules with separate responsibilities.

The main processing module is responsible for the trace file identification, processing and storage of trace files given as an input. Furthermore, it is responsible for loading the data of trace file that is stored to the local database and update changes to its content.

The Metafile and Sub metafile modules are responsible for handling all the metafiles and sub metafiles that stored in the metafile and sub metafile repository. They are responsible for loading them and checking their validity. The metrics module is responsible for producing all the QoS measurements and the Charts module for plotting a chart requested by the user.

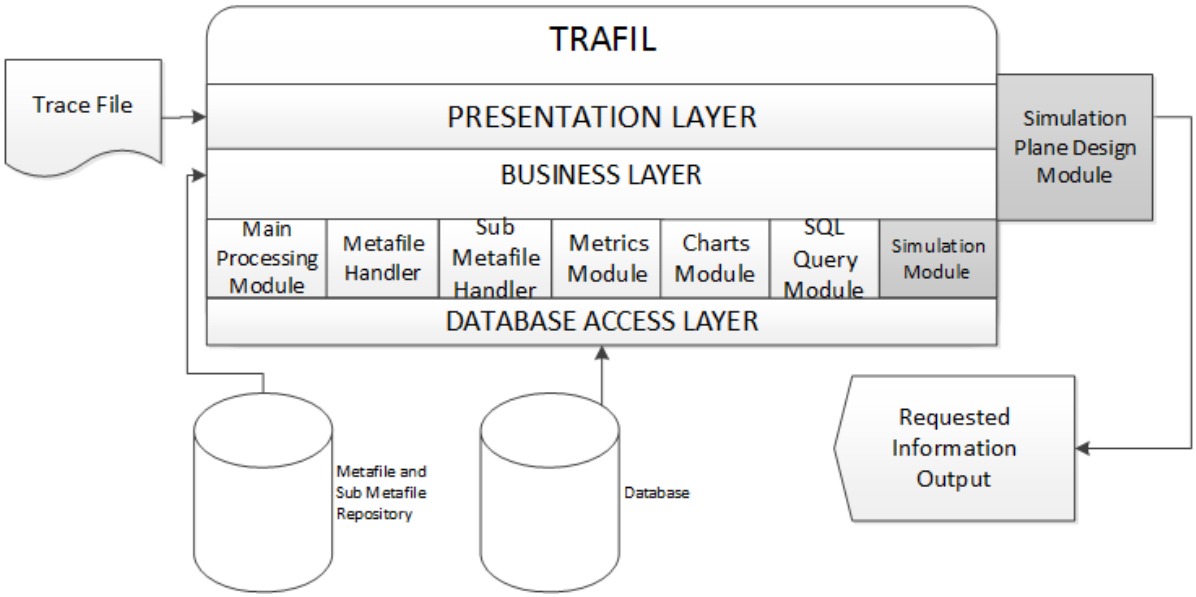

Figure 1: TRAFIL Architecture.

The user initiated SQL queries to the local database are handled by the SQL Queries module and the execution of Video transmission simulations using the Evalvid-RA framework by the Simulation Module.

Figure 1 also depicts the new modules added to TRAFIL which include the Simulation Plane Design module. This new module enables the user to design the simulation using a graphical interface alleviating this way the task of having to construct an OTcl script. TRAFIL creates the script according to user input and can then execute it.

Metafiles and sub metafiles (Bouras, 2013) serve as utility files and are used by TRAFIL in various instances including the trace file (NS-2 Trace Formats, 2013) identification procedure. Metafiles describe the structure of a trace file, they contain all the different fields a trace file logs and additional information that serves a variety of purposes.

Each metafile encodes the structure and content of a line regarding a specific trace file format. They are used by TRAFIL primarily to identify a trace file that has been given as an input from a user. Following the identification, TRAFIL uses the metafile that has been matched with the trace file to process its contents and finally to store it to the local database.

In addition to the metafiles, TRAFIL employs sub metafiles. The sub metafiles are also used as utility files that complement the use of the metafiles just like NS-2 uses extra headers for each main trace file format depending on the packet type. In order to identify and process these extra headers that might be present in each trace file format TRAFIL uses in the majority of cases a sub metafile for each header that is supported by NS-2. More information about metafiles and sub metafiles can be found in (Bouras, 2013).

TRAFIL supports a variety of features that aim to aid users throughout the post-simulation analysis procedure of trace files. Firstly, TRAFIL can be used to parse and process all the different trace file formats that are produced by NS-2 (Normal, Old Wireless, New Wireless). This flexibility is accomplished due the use of metafiles and sub metafiles.

Trace files that have been identified and parsed by TRAFIL can then be subjected to analysis operations. TRAFIL facilitates the analysis operations by enabling users to calculate a variety of Quality of Service (QoS) measurements regarding a simulation. These measurements include Throughput (bits / second), Packet Delivery Rate (packets/second), Minimum End to End Delay (seconds), Maximum End to End Delay(seconds), Average End to End Delay (seconds), Delay Jitter, Minimum Delay Jitter (seconds), Maximum Delay Jitter (seconds), Average Delay Jitter (seconds) and Packet Loss Ratio (packets/seconds). In addition to the QoS measurements TRAFIL gives the ability to plot charts regarding the simulation and the traffic between either two specific nodes or a single node of the simulation. The charts that TRAFIL supports are: Packet Delivery Rate (packets/second), Throughput (bytes/second), Delay Jitter (seconds) and Packet Loss Ratio (packets/second). Finally, TRAFIL calculates all the general simulation information e.g. Start and End Time, Number of Nodes, Number of Packets Dropped, Number of Packets Sent, Number of Packets Forwarded. The General Simulation Information is calculated both for the simulation as a whole and for specific nodes depending on user selection.

Moreover, TRAFIL supports the execution of video transmission simulations using NS-2 and the Evalvid-RA framework. The Evalvid-RA module is very popular among researchers and has been used extensively for simulating the transmission of a video file across a variety of networks. Nevertheless, the setup and parameterization of the tools that accompany Evalvid-RA can be a wearing task. Therefore, TRAFIL enables users to execute video transmission simulations and process the results via an easy to use user interface, given that the user has successfully installed NS-2 and incorporated the Evalvid-RA files.

3 SIMULATION PLANE DESCRIPTION

TRAFIL is a tool that used to mostly facilitate the post-simulation analysis of simulation results (trace files). It succeeded firstly in offering an easy to use framework that presented a more abstract approach towards parsing and identifying trace files and making this process significantly faster compared to alternative approaches (Bouras, 2013). Furthermore, TRAFIL presented some novel features that included storing each trace file to a local database but the most important one is the opportunity to conduct video transmission simulations using NS-2 and the Evalvid-RA module.

The feature of supporting video transmission simulations was a first step towards automating the procedures of designing, executing and finally analysing a simulation and its results. Nevertheless, the simulation scenario used for executing simulations still had to be created using the OTcl scripting language and had to be prepared in advance by the user.

Thus, the design and preparation phase of the simulation still required the user to be able to code in OTcl and be familiar with NS-2 parameters and objects.

Therefore, we recently introduced to TRAFIL a new functionality that enables users to design the simulation plane using a graphical user interface. Through this GUI, users can select all the network components that make up a simulation and place them inside the simulation plane. These components include network nodes, network links, traffic generators, network agents, applications etc. Each object can be selected from a palette and after its placement inside the simulation plane it can be edited and parameterized in order to perform as the user expects. All the network components are compatible with the ones that NS-2 supports since after the user has finished designing the simulation, it is translated by TRAFIL to an OTcl script in order to be executed by NS-2.

In this section we present the architecture of this new module and thoroughly explain its purpose and what it aims to accomplish.

3.1 Simulation Plane

The simulation plane feature enables the user to describe a network in a way that is closer to designing rather than programming. Figure 2 presents its architecture which is organized in 4-layers.

The first is the design layer. A key part of this layer is the simulation panel; a design environment similar to the layout is used in most design programs. It consists of the palette, the design panel, and several pop-up menus.

The palette includes all input options for the simulation panel, such as all node types, a links list, and buttons that control many network parameters. The rest of the simulation plane (centre and right area) consists of the design panel. The drawing panel is where the network formation will be painted. Depending on the "painting tool" selected (node, link) a certain shape will be painted there, resembling a network component. That shape will often give access to all the menus related to that component.

All menus appear in separate windows, and allow for node customization, or choosing links between already existing nodes, or specifying the parameters needed to the simulation that are not part of the visual design. Those parameters can be simulation times, file names etc. The user can keep these windows open while still using the drawing surface to edit or add more features to his network.

The script layer can only follow the design layer. It receives the data created in the previous layer, and processes them in order to create the simulation file,

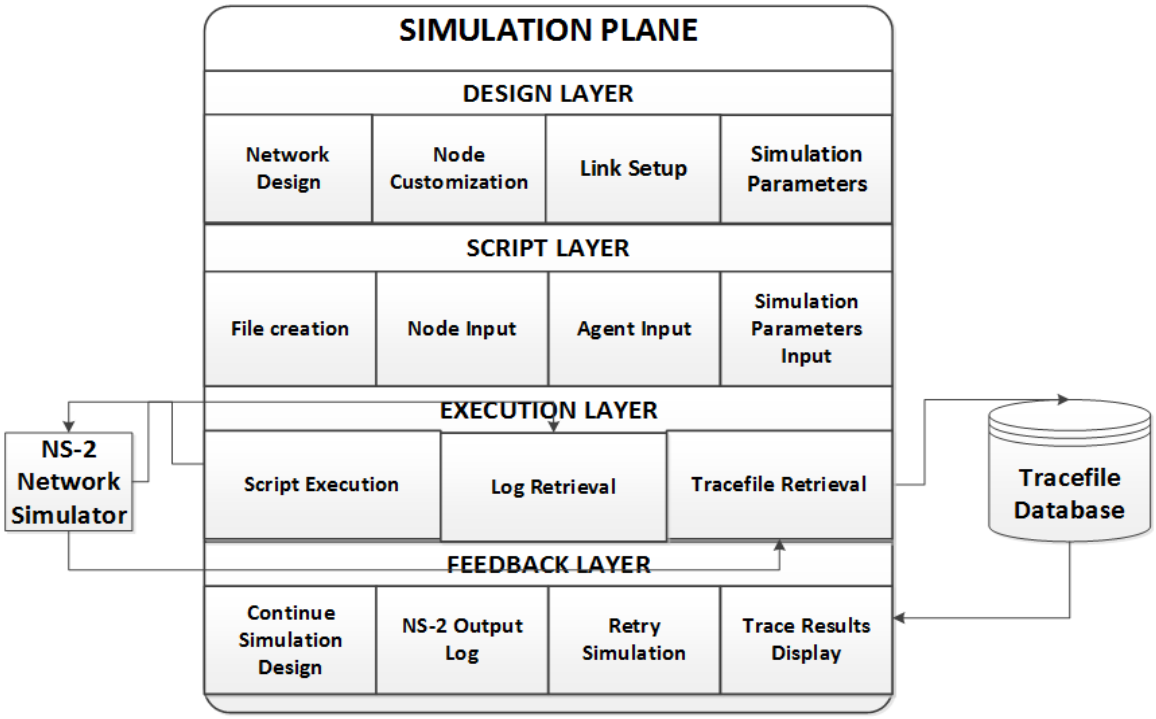

Figure 2: Simulation Plane Architecture.

according to OTcl language's rules.

The next layer is the execution layer. The simulation script file is the input of this layer which is then forwarded to NS-2, using the required syntax to perform the simulation. After the simulation is complete, the execution layer retrieves the output log as well as the files produced by NS-2, and sends them to the next layer. The resulting trace file is automatically loaded to TRAFIL's database.

The last layer is the feedback layer. Here TRAFIL uses a window which contains options for displaying all the data received from the execution. Moreover, there are options to rerun the same simulation again or return to the design layer.

3.2 Node Creation and Configuration

One of the most important components of a Network Simulation is a node. A node can represent a variety of entities but its importance lies in the fact that it is the means for introducing data traffic in a network.

NS-2 supports both wired and wireless nodes, each with its own unique parameters. Therefore, TRAFIL enables users to create any of these node types via the graphical user interface. The node is selected from a palette and can be either wired or wireless. The selected node can be placed and dragged anywhere inside the drawing panel.

TRAFIL's new module gives the opportunity to users to create a whole network pattern of nodes, and equip them with any available options, using a pop-up menu designed for easy node parameter configuration. In order to ease this process, we have chosen to avoid manual entry as much as possible, using drop-down menus and pre-set values. This way, the user can keep control of his designed network without having to deal with OTcl code.

All the node parameters can be edited and correspond to the ones that can be set using OTcl. The difference lies in the fact that the user is not obligated to know beforehand the commands required to set them, TRAFIL gives the ability to simply enter the appropriate value for the desired parameters. In order to edit the parameters of a specific node the user can simply double-click the node on the simulation plane, spawning an edit window inside which are all the parameters.

Each node can be re-edited in subsequent simulations i.e. change position in the simulation plane, parameters, identification number. Furthermore, the simulation plane along with the node parameters can be saved and re-loaded in order to be re-used.

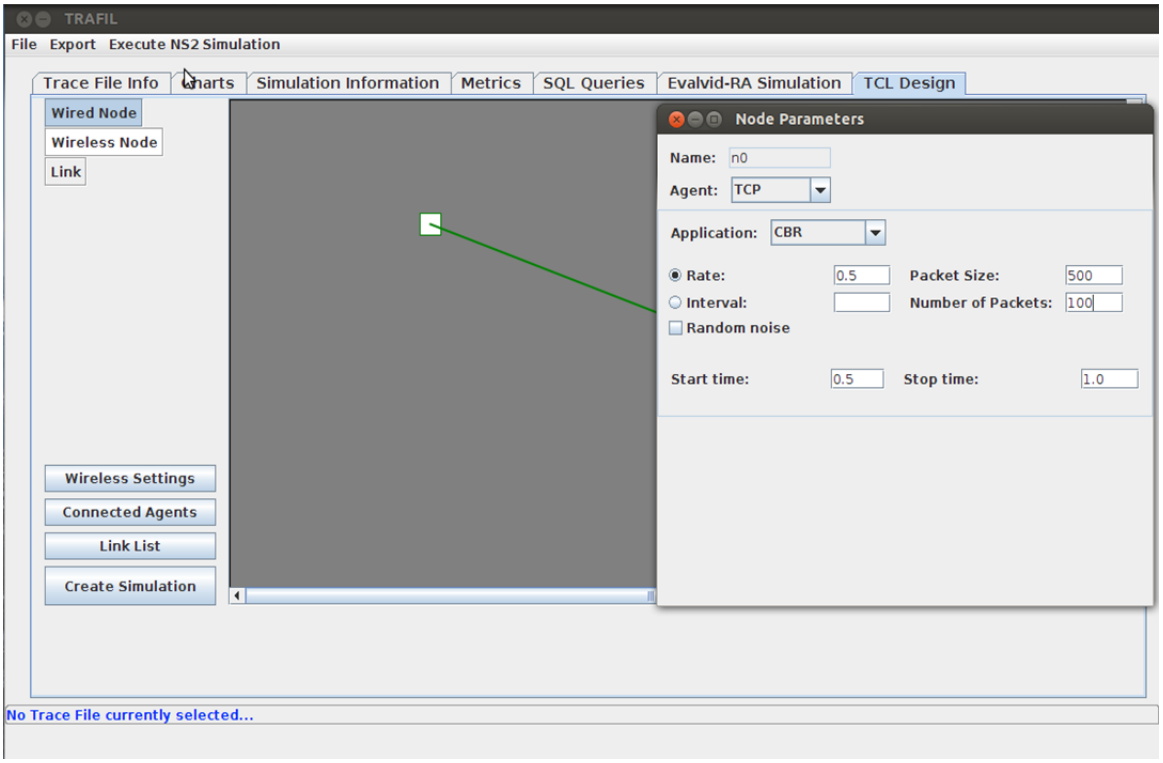

Figure 3: Configuring node parameters.

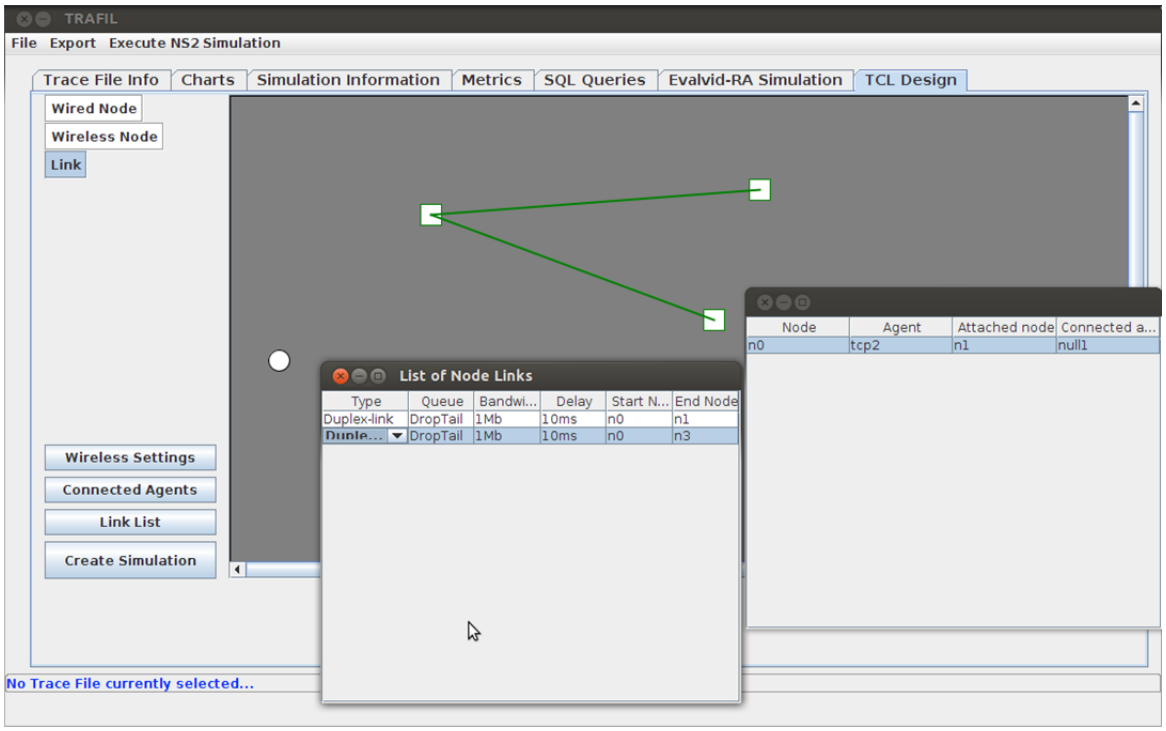

Figure 4: A simple network design.

3.3 Agents-Traffic Generators-Applications

In the previous sub section we mentioned simulation nodes and their importance for a simulation. Nodes are indeed the means with which traffic is introduced in the simulated network but in order for a node to send packets agents and traffic generators must be used.

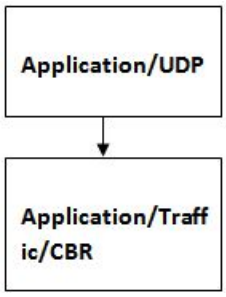

Figure 5: Agent and Application connection.

Figure 5 llustrates how agents and applications are connected in order for traffic to be created. NS-2 supports a variety of agents (sending and receiving) as well as traffic generators. The traffic generators can also be either user defined or simulated like the one in 5.

TRAFIL enables a user to use most common agents or traffic generators supported by NS-2. TRAFIL currently supports all NS-2 traffic generators. Also, the protocol agents currently supported are TCP, User Datagram Protocol (UDP) and Null. The selection of the agent and traffic generator can be made via the parameter window of each node since an agent along with his traffic generator must be applied to a node. Each of those selections (if applicable) displays its own parameters in the same window, with pre-set values for the most common properties, or values the user previously inputted.

3.4 Links

In wired scenarios all the nodes of the simulation that must communicate and exchange information have to be connected using links. NS-2 simulations require the user to define each link between the nodes along with each link type's parameters. There is a variety of links available by NS-2 which are supported by TRAFIL.

In order to establish a link between two nodes on the simulation plane the user chooses the appropriate button from the palette menu, entering this way the linking mode. While in the linking mode, each click on an already existing node notes a link ending. As suggested, with every second click a link is closed, appearing on the design panel. It is also listed in the

link list, which can be viewed anytime via the corresponding palette button. Through that same menu, a user can easily define link parameters such as its type, queue, and the nodes it's attached on. The user can also delete existing links using that menu.

3.5 Simulation Design Example

Figure 4 depicts the Graphic User Interface of TRAFIL's design module. In the example shown, 3 wired nodes (red squares) are set up and linked together, while another wireless node (blue circle) is present. All the links are listed in the open window which also shows all the parameters of a link, allowing for their customization.

4 SIMULATION EXECUTION

Although TRAFIL was originally created as a post-simulation front-end framework, many pre-simulation features have since been added. For all those pre-simulation features to work, a function calling the NS-2 simulator was created, allowing the user to execute any OTcl script through TRAFIL graphical user interface. The results of that simulation are automatically imported in the local trace file database, allowing direct and seamless access to the results.

This same function is a key to the new module of simulation design and script generation. It is used by TRAFIL in order to commence any NS-2 and get the results in a fashion similar to the function's normal use. The exact procedure is described below.

4.1 Tcl File Creation

Following the process described in Section 3, the user has already prepared a network design and all of its properties as desired. After the design is complete, a script generation button placed in the design palette starts the procedure of creating the Tcl file. That button brings up a menu that asks the user for some final details relevant only to the script file itself, and not to the network. For example, the user can define specific file names or enable TRAFIL supported NS-2 add-ons. This menu also specifies the execution times, such as simulation begin/end, pauses, etc.

An OTcl script such as the one TRAFIL will generate consists of several parts. First, some basic lines that exist in every NS-2 OTcl script are written, with pre-defined or user-defined (if there is such user input) options. Those options can be customized via the script generation button menu described above. They can include all the file names that will be used, along with the option of enabling nam (in case the user needs that output file for his own reasons) or any other NS-2 add-on supported by TRAFIL.

The execution time parameters (such as simulation begin-end and any pauses required) are also input in this menu.

Second, the node table is written. This step is pretty straight-forward; all the nodes on the design panel are being mapped on a node table, along with their parameters (described in each pop-up menu provided with every node), and that table is used to generate the node creation script lines. That table is vital in creating the rest of the script as well.

After that, any links required by the user that are listed in the corresponding list described in section 3 are tabled, and passed in the script along with their parameters.

With the basic network layout ready, agent information is written. This information is pulled from the node table mentioned above, in the node creation lines. Any parameters mentioned in the node tables are translated into NS-2 valid parameters, and written in the script.

Finally, after establishing connections between nodes, the simulation time parameters are written, pulled from the script generation button menu described above.

4.2 Script Execution

If the Tcl file creation is successful, TRAFIL saves the script in a folder dedicated for that purpose. Then, the user is given the option to run it right away through TRAFIL's innate NS-2 script execution function. The results will be directly imported into TRAFIL's database, as described in previous sections. However, if the Tcl file creation was not successful, e.g. if any parameters were missing or if the network layout was incomplete, NS-2 output will be shown in a report window according to the script execution function's order, informing the user about the errors.

5 CONCLUSIONS FUTURE WORK

We presented in this work a new module of TRAFIL. The innovation of this lies with the fact that it provides an easy to use tool, whether it is a novice

NS-2 user, or a veteran one, which covers all aspects of simulation from start to end. With most of the common tasks simplified and completed by the program itself, he can simply focus on what's important: The simulation and the test results, while generating error-free scripts. The user may also use additional TRAFIL functionality such as simple NS-2 simulation and result output, chart creation, important variable metrics and custom calculations in the form of SQL Queries.

Our conclusions are that the pre-simulation design and execution enhancements significantly increase TRAFIL's potential, since they can now be combined with its metafile-based flexibility and form a powerful, expandable and easily accessible simulation framework. Furthermore, simulation design process can be reversed since existing NS-2 scripts can be recognised and loaded within the framework for further processing.

Our future work will include extensions in the simulation design module that will enable a more complete support of NS-2 functionalities and add-ons. We also plan to investigate the possibility of generalizing the framework to operate around other simulators such as NS-3, in order be able to provide a generic simulation facilitation framework.

REFERENCES

Breslau, L., Estrin, D., Fall, K., Floyd, S., Heidemann, J., Helmy, A., Huang, P., McCanne, S., Varadhan, K., Xu, Y., Yu, H., 2000. *Advances in Network Simulation*, *IEEE Computer*, Vol. 33, No. 5, pp. 59 – 67.

Fujimoto, R., Perumalla, K., Park, A., Wu, H., Am-mar, M., Riley, G., 2003. *Large-scale network simulation: how big? how fast? Modeling, Analysis and Simulation of Computer Telecommunications Systems,* 2003. MASCOTS 2003. *11th IEEE/ACM Interna-tional Symposium* on, pages 116-123, 2003.

Weingärtner, E., vom Lehn, H., Wehrle, K., 2009. *A performance comparison of recent network simulators*. In *Proceedings of the IEEE International Conference on Communications* 2009 (ICC 2009), Dresden, Germany, IEEE.

Azizur Rahman, M., Pakštas, A., Zhigang Wang F., .2009. *Network modelling and simulation tools, Simulation Modelling Practice and Theory*, Volume 17, Issue 6, July 2009, Pages 1011-1031, ISSN 1569-190X, 10.1016/j.simpat.2009.02.005.

Wang, S.Y., Chou, C.L., Lin, C.C., 2007. *The design and implementation of the NCTUns network simulation engine, Simulation Modelling Practice and Theory*, Volume 15, Issue 1, January 2007, Pages 57-81, ISSN 1569-190X, 10.1016/j.simpat.2006.09.013.

Network Simulator NS-2 website: http://www.isi.edu/nsnam/ns/ (Accessed May 2013).

Cicconetti, C., Mingozzi, E., Stea, G., 2006. *An Integrated Framework for Enabling Effective Data Collection and Statistical Analysis with ns-2*. *In Proceedings of WNS2* '06; *Proceeding from the 2006 workshop on ns-2*: the IP network simulator.

Chen, Q., Schmidt-Eisenlohr, F., Jiang, D., Torrent-Moreno, M., Delgrossi, L., Hartenstein, H., 2007. *Overhaul of ieee 802.11 modeling and simulation in ns-2*, *Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, October 22-26, 2007, Chania, Crete Island, Greece.

NS-2 Contributed Code: http://nsnam.isi.edu/nsnam/index.php/Contributed_Code#Support, (Accessed May 2013).

Luis Font, J., Iñigo, P., Domínguez, M., Luis Sevillano, J., Amaya, C., 2011. *Analysis of source code metrics from ns-2 and ns-3 network simulators, Simulation Modelling Practice and Theory*, Volume 19, Issue 5, May 2011, Pages 1330-1346, ISSN 1569-190X, 10.1016/j.simpat.2011.01.009.

NS-2 Trace Formats: http://nsnam.isi.edu/nsnam/index.php/NS-2_Trace_Formats (Accessed May 2013).

Freitag Borin, J., L.S. da Fonseca, N., 2008. *Simulator for WiMAX networks, Simulation Modelling Practice and Theory*, Volume 16, Issue 7, August 2008, Pages 817-833, ISSN 1569-190X, 10.1016/j.simpat.2008.05.002.

Bouras, C., Charalambides, S., Drakoulelis, M., Kioumourtzis, G., Stamos, K., 2013. *A tool for automating network simulation and processing tracing data files. Simulation Modelling Practice and Theory* Vol 30, 2013, Pages 90-110.

TRAFIL download: http://ru6.cti.gr/ru6/research_tools.php#TRAFIL (Accessed May 2013)