Enhancing simulation environments with TRAFIL

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

[1]Computer Technology Institute and Press, N. Kazantzaki Str, University Campus 26504 Rio Greece
[2]Computer Engineering and Informatics Department, University of Patras
[3]Center for Security Studies, P.Kanellopoulou 4, T.K 10177, Athens, Greece
{bouras, stamos}@cti.gr, {charalampi, drakouleli}@ceid.upatras.gr, g.kioumourtzis@kemea-research.gr

ABSTRACT

This chapter presents TRAFIL, a comprehensive tool for enhancing execution of simulations. It provides an overview of the tool, its architecture and its functionalities. It explains how TRAFIL enhances the entire simulation procedure including graphical setup of simulation scenarios, automated execution of simulations, flexible handling and storage of simulation trace files and presentation of plots based on processing of simulation results. It presents the concept of metafiles that provides TRAFIL with the flexibility to handle heterogeneous simulation environments. The chapter also compares TRAFIL performance with other similar tools and finds that it offers significantly improved performance. It therefore concludes that TRAFIL offers both a rich set of simulation enhancement functionalities and top performance.

1. INTRODUCTION

This chapter presents TRAFIL (TRAce FILe [1]), a comprehensive tool for enhancing execution of simulations. It wraps the entire simulation procedure from scenario setup to results analysis, including graphical setup of simulation scenarios, automated execution of simulations, flexible handling and storage of simulation trace files and presentation of plots based on processing of simulation results. TRAFIL is based on the NS-2 (Network Simulator), but is built on an extensible architecture that allows any NS-2 additional plug-in to be configured and supported, and can even be extended to process trace files from other simulators.

The chapter focuses on TRAFIL architecture and features and intends to serve both as an introduction and presentation of the tool, and also as a general discussion of processing techniques for simulation results, by detailing the reasoning for the choices made in the tool architecture. It also introduces the design considerations that enable the tool to be extensible, which may be useful for building flexible and usable tools around existing environments.

TRAFIL aims to make the execution of a great number of network simulations quicker, and the extraction of results from a large amount of data more flexible and productive. It offers the possibility to design, create, execute and review NS-2 simulation scenarios, and also offers post-simulation trace analysis functionalities. It is therefore 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.

In order to accomplish the post-simulation tasks, TRAFIL presents a novel way of interpreting, parsing, reading and eventually using NS-2 trace files. It introduces the notion of “metafiles” and “sub metafiles” throughout the procedures of trace file recognition and parsing, making the overall analysis operation substantially efficient and faster than alternative approaches. Metafiles and sub metafiles are used to encode NS-2 trace file structures enabling a more abstract approach to the trace file processing operation. Furthermore, TRAFIL facilitates the overall trace file analysis task by offering the opportunity to store each trace file as well as every Quality of Service (QoS) measurement produced for each trace file. Following the trace file recognition and processing operations, the information contained in a trace file is presented through a Graphical User Interface (GUI) offered by TRAFIL along with a variety of data, metrics and statistics related to simulation results. Finally, the tool offers the opportunity to execute custom Structured Query Language (SQL) queries to the local database and to completely automate the simulation procedure by enabling the user to execute NS-2 scripts as well as perform a simulation of a video transmission using the Evalvid-RA framework.

The rest of this chapter is structured as follows: Section 2 presents related work for other similar network simulation enhancement tools. Section 3 gives an overview of TRAFIL and section 4 focuses on the metafiles concept. Section 5 presents in detail the graphical simulation setup capabilities of the tool, section 6 its trace file analysis functionalities, section 7 its simulation execution wrapper and section 8 the way that TRAFIL can present simulation plots and results, as well as its performance compared to other similar tools. Section 9 summarizes our conclusions and section 10 closes the chapter with our future work plans.

2. RELATED WORK

Similar work [2] and tools have been developed that produce statistics of a simulated network's behaviour. Some of these projects ([3], [4]) have integrated NS-2 unlike TRAFIL which uses NS-2 trace files to produce the requested statistics. Also there are tools like JTrana, Trace Graph and NS-2 Trace Analyzer which offer the opportunity to analyze NS-2 trace files by producing numerous statistics, measurements and charts.

Trace Graph ([5], [6]) is an NS-2 trace file analysis tool. This tool provides many options for analysis, including a variety of charts and statistical reports. It is implemented in MATLAB 6.0 [7] and can be compiled to run without MATLAB. This tool also gives the user the ability to extract from a given NS-2 trace file useful statistics through a graphical user interface. The kind of statistics that can be obtained include node statistics, network statistics and QoS metrics. It also produces 2D and 3D graphs for measurements like cumulative sums, throughput, throughput vs. delay, jitter, packet ID's and other common statistics. Finally, this tool supports the following NS-2 trace file formats: old wireless, new wireless, wired.

JTrana [8] is a Java based NS-2 wireless trace analyzer. It can be used to analyse the NS-2 wireless simulation trace files through a GUI. Features of JTrana include production of overall network information and plotting of numerous charts regarding that information. JTrana supports both wired and wireless trace files and uses a MySQL database to store the trace file that is subject to analysis at a given time.

NS-2 Trace Analyzer ([9], [10]) is a command line tool written in C/C++ and is designed for use in all OS platforms and Cygwin. As the previous analysis tools, NS-2 Trace Analyzer can be used to obtain common network statistics using the trace file from a simulation. This tool does not offer the opportunity to create charts regarding the statistics that the user retrieves about a simulation.

The aforementioned tools provide useful information regarding only a specific simulation scenario and therefore all the metrics and results refer to only one trace file. Furthermore, these tools do not provide the user with the opportunity to store each trace file he has analyzed locally, for instance in a database, so that he can reuse it without having to reopen it. In order to extract information regarding another simulation the user has to load another trace file. This is a rather slow task and it can be acceptable for small sized trace files but when it comes to simulations that produce large trace files this process can be considerably time consuming. Also, in order to alter the contents of a trace file and compare the results with a previous analysis a user has to reload the trace file every time. Adding to this is the fact that the results of the analysis that a trace file is subjected are only saved to text files. It is up to the user to keep them organized and safe so that he can be able to reuse them.

In terms of performance when it comes to loading a trace file the earlier mentioned tools behave well for small sized trace files. Although, when it comes to serious simulations that produce trace files in the orders of MB's the performance deteriorates significantly.

Finally, there is J-Sim ([11], [12]) a Java based open source, component-based simulation environment based on the idea of the Autonomous Component Architecture (ACA). As a result, components are one of the basic entities of J-Sim and they can be individually designed, implemented and tested. Also the way components interact and act in regard to the data transfers is specified at system design time. J-Sim as is the case with TRAFIL is platform independent due to the programming language it is implemented in. In addition, J-Sim can be used in conjunction with scripting languages like Tcl or Python. The scripting languages are used to combine and hold together in one sense the different Java classes as it is done with NS-2 C++ classes and OTcl. In order for a simulation to be created the basic package is the drcl.inet which contains the base classes defined in the abstract network model, as well as a set of utility functions and classes that facilitate creation of simulation scenarios.

Furthermore, except of the TCL/Java scripting that a user can incorporate to create and orchestrate a simulation, J-Sim can be used in combination with gEditor. gEditor is a graphical user interface that serves as a front for J-Sim enabling the user to create the simulation plane without using TCL/Java, gEditor takes up the responsibility to interpret the parameters and objects that have been requested and create the corresponding objects. It passes the appropriate components to J-Sim via its console and runs the whole simulation on behalf of the user. This is a very useful feature that gives the ability to rapidly and conveniently create and execute a simulation.

As it will be shown in the following sections TRAFIL aims to offer a similar feature but is targeted though towards NS-2. TRAFIL enables the user to input a script that describes the simulation plane through a graphical user interface. The description and parameters are used to produce trace files which are eventually subjected to analysis akin to the procedures of the aforementioned post simulation analysis tools. In a few words, the final objective is to succeed in offering a tool that can be used to perform a complete simulation and its analysis in a flexible, effortless and robust manner.

3. TRAFIL OVERVIEW

TRAFIL is a tool that can be used to automate the complete network simulation procedure. Namely, it offers the ability to graphically design and execute a network simulation as well as analyze its results.

TRAFIL supports network simulations by utilizing NS-2 and it comes with out-of-the-box support for designing NS-2 simulation scenarios, execution of these scenarios by invoking NS-2 and analysis of the produced trace files.

One of TRAFIL's most important features is that it disengages the user of the effort to learn NS-2's specifics in order to design a simulation. Although it offers the ability to execute an NS-2 OTcl simulation script (OTcl is the scripting language used for writing NS-2 simulations scripts), TRAFIL enables users to design the simulation plane using a graphical user interface (GUI). Via this GUI, users can select all network components that make up a simulation and place them inside the simulation plane.

Moreover, in terms of post-simulation analysis, TRAFIL is not strictly bound to NS-2 trace files. Although TRAFIL supports every different trace file produced by NS-2 it can also be extended to support a variety of trace files. This is accomplished by creating custom *metafiles* (presented in detail in the next chapter) that describe a new trace file's structure. Metafiles were introduced in the post-simulation analysis domain by TRAFIL in order to offer a more abstract approach to processing simulations' results. They describe a trace file's structure and are used by TRAFIL to mine specific information required for producing charts and measurements.

Finally, TRAFIL executes the simulation scenario by invoking NS-2. However, TRAFIL also offers the ability to simulate video transmission scenarios by utilizing the Evalvid-RA framework [13] [14]. Evalvid-RA is an added module to NS-2 which is based on the generation of a trace file (regarding a video file) and can support its rate-adaptive multimedia transfer. In order to use Evalvid-RA along with NS-2, specific pre-processing and post-processing steps are required before and after the NS-2 simulation respectively, that involve the usage of certain tools for video processing. TRAFIL automates the video transmission simulation procedure by incorporating Evalvid-RA's utility tools. Therefore, the user is able to define all the parameters as he would have done when executing a normal Evalvid-RA simulation procedure, and specify the simulation script he wants to test. TRAFIL carries out the simulation procedure returning its results in a user friendly manner.

This section gave a brief introduction to TRAFIL's main operations. The next section presents TRAFIL's actual architecture and discusses each component's role in performing TRAFIL's main operations.

3.1 TRAFIL Architecture

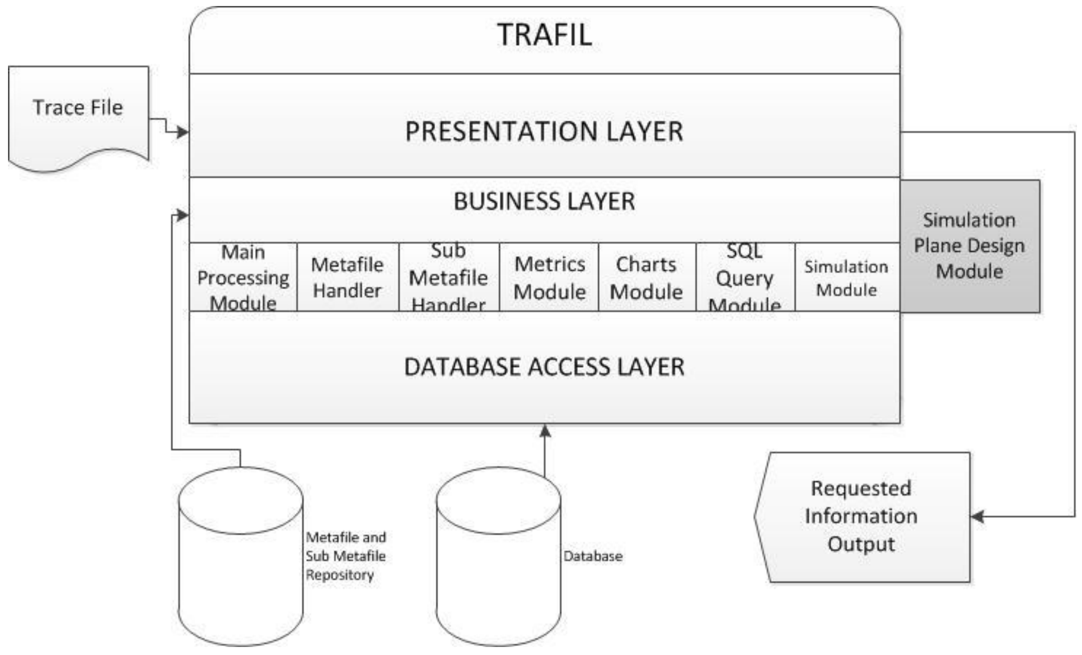


Figure 1 TRAFIL Architecture

TRAFIL's architecture follows a three tier model as can be seen in Figure 1 and the three basic tiers are: Presentation layer, Business layer and Database Access layer.

The presentation layer introduces the user to TRAFIL's functionality and is depicted in Figure 2. Via the presentation layer the user can invoke any utility offered by TRAFIL given that there is a trace file loaded. A trace file can be loaded either by opening a new trace file from the file system or by loading a pre-existing one from TRAFIL's database. As shown in Figure 2 no trace file is currently selected and thus no data are shown.

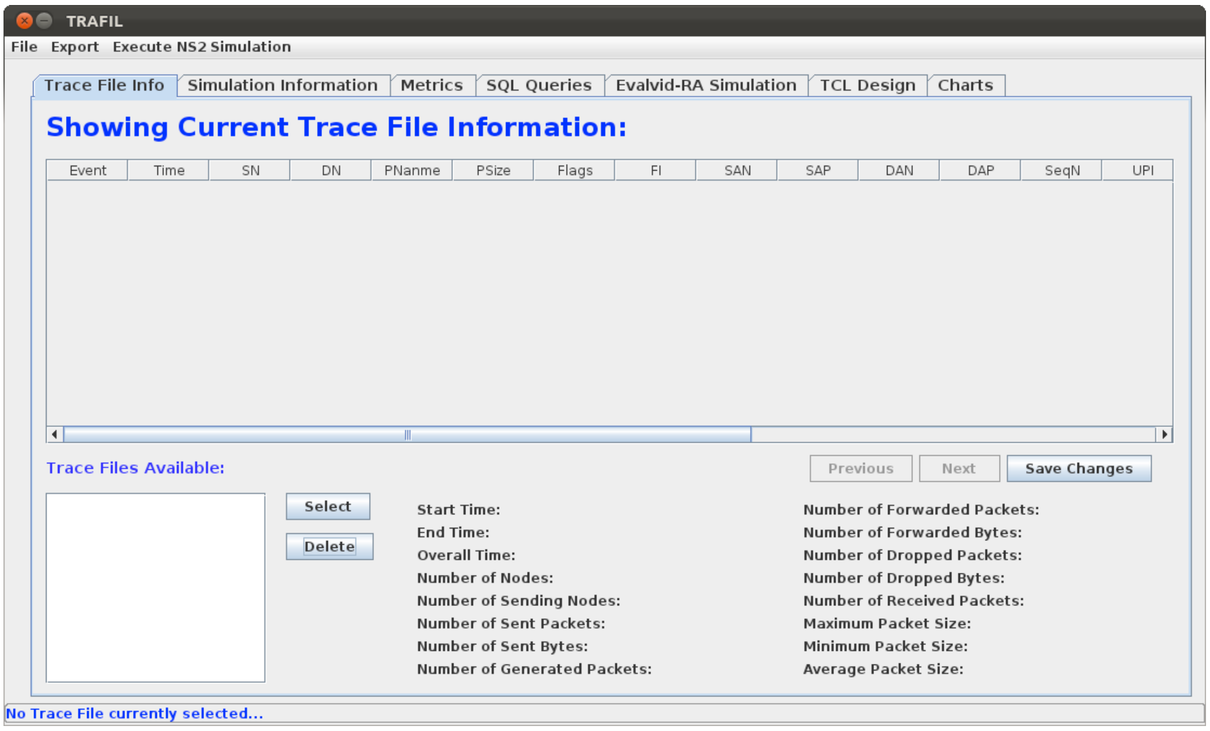


Figure 2 TRAFIL Presentation Layer

Every user request is carried out in the business layer and the results are returned and depicted at the presentation layer. As shown in Figure 2 TRAFIL can produce Metrics and Charts (at the appropriate tabs) based on the trace file information as well as general simulation information that summarizes the events that took place in the specific simulation scenario. The most important part of the general simulation information is also depicted in TRAFIL's Trace File Info tab along with the selected trace file's first 50,000 lines. Moreover, via the SQL Queries tab users can issue queries directly to TRAFIL's database if they require more specific measurements that are not currently supported by TRAFIL. Finally, TRAFIL supports the novel feature of graphically designing simulation scenarios via the TCL tab and their execution via the Execute NS-2 Simulation menu item. Finally, another unique feature is the ability to execute video transmission simulations using Evalvid-RA in conjunction with NS-2 via the Evalvid-RA Simulation tab.

The aforementioned TRAFIL features are handled by distinct business layer modules as shown in Figure 1. The core module is the Main Processing Module and is responsible for the identification of the selected trace file's type. Namely, it determines whether it is a Normal, Old Wireless, New Wireless or a user specified trace file. Moreover, it is responsible for parsing, processing and storing the trace file's data to TRAFIL's database if it is a new trace file or load its data from the database if it is a pre-existing one. The Metafile and Sub-Metafile modules, as their name suggests, are responsible for providing an interface to other modules that need to access metafile data for their operations i.e. during the trace file parsing and identification procedures as we mentioned before. The Metrics module is responsible for calculating the following QoS measurements: Packet Delivery Rate, Throughput, Minimum End to End Delay, Maximum End to End Delay, Average End to End Delay, Delay Jitter, Average Delay Jitter, Minimum Delay Jitter, Maximum Delay Jitter, Packet Loss Ratio. The Charts Module plots specific charts that refer to either specific simulation Nodes or the communication between two Nodes. The SQL Query module is responsible for validating and executing the user issued SQL queries. Finally, the Simulation and Simulation Plane Design Modules support TRAFIL's two novel features. The latter is responsible for the operations required for producing a valid NS-2 simulation script based on the graphical design of the simulation plane. The former is responsible for handling either the execution of a simple NS-2 script that is specified by the user or a video transmission simulation. In the video simulation case it is responsible for coordinating the execution of a series of steps that involve using external programs that process the input video based on user specified parameters.

The Database access layer interacts solely with the Business layer and is responsible for storing data to the local database or returning the data requested by the Business Layer.

Finally, as Figure 1 depicts TRAFIL uses a MySQL database to store each trace file’s data. The database holds each trace file’s data in a distinct table and its design is optimized for faster data transfer and retrieval. As shown in [15] TRAFIL manages to outperform similar tools in terms of trace file processing and storage speeds. Moreover, TRAFIL has a metafile and sub metafile repository that contains the actual metafiles and their sub metafiles. Each metafile's sub metafiles are stored in a unique folder named after the metafile they belong to. This way the loading of the sub metafiles for a specific metafile becomes simpler. Thus, in the case that a user wants to introduce his own metafile this convention should be followed in order for TRAFIL to support it. Namely, the metafile should be added to the repository and its sub metafiles (if any), to unique folder named after the metafile.

4. METAFILES

Metafiles have been used in various applications to describe the structure or the content of another file. The most useful aspect of metafiles is the fact that they render the applications that use them more generic or abstract as they become independent of the content or the format of the input. It is evident that the use of metafiles is indeed popular and causes applications to be more robust and adaptable, but all trace file analysis tools until now haven't made any use of them. On the contrary they encode the structure of a trace file internally. TRAFIL on the other hand makes use of metafiles during the trace file parsing, processing, analysis and storage procedures.

In order to identify and analyze a trace file there must be a way to know its structure and to expect in some degree the input. That is why TRAFIL uses metafiles. Metafiles encode the structure of each different trace format produced by NS-2, they contain information about the number of fields, number of columns, the names of each column and what data type each column is. In other words they contain all the necessary information to describe the data of a trace file.

Using each metafile TRAFIL is no longer dependant of the structure or the data types of an input. If there is a metafile that is constructed correctly so that it can describe a specific trace file format the tool will be able to process it. TRAFIL is not even dependant on the number of metafiles, if there is not a metafile present which can be matched to an input TRAFIL will acknowledge it and report the issue so that the user will construct the appropriate metafile.

The structure of a metafile is depicted in Figure 3. As Figure 3 shows there are a number of different fields contained in a specific metafile. The metafile in the figure below is the one that encodes the structure of a Normal trace file and it is utilized when TRAFIL needs to process a trace file of that format. The first three fields are mandatory for every metafile and are extremely important in the trace file recognition phase. The *NumberOfColumns* element is used to demonstrate the actual number of elements that are present in each line read from a trace file of Normal trace format and are separated by white spaces.

```
NumberOfFields 14
NumberOfColumns 12
UniqueCounter 1
-name event -type char(1) -index 0 -unique +
-name time -type double -index 1
-name SourceNode -type int -index 2
-name DestinationNode -type int -index 3
-name PacketName -type varchar(20) -index 4
-name PacketSize -type int -index 5
-name Flags -type varchar(7) -index 6
-name DestinationAddress -type int -index 9 -delimiter .
-name FlowID -type int -index 7
-name SourceAddress -type int -index 8 -delimiter .
-name SourcePort -type int -index 8
-name DestinationPort -type int -index 9
-name SequenceNumber -type int -index 10
-name UniquePacketID -type int -index 11
-TimeRelated -column time
-NodeRelated -column SourceNode -column DestinationNode
-PacketSize -column PacketSize
-SendingNodes -column SourceNode -column SourceAddress
-GeneratedPackets -column SourceNode -column SourceAddress -column UniquePacketID
-ReceivedPackets -column DestinationNode -column DestinationAddress -column UniquePacketID
-ForwardedPackets -column SourceNode -column SourceAddress -column DestinationAddress -column UniquePacketID
-SentPackets -column SourceNode -column SourceAddress -column UniquePacketID
-DroppedPackets -column UniquePacketID
```

Figure 3 Metafile Structure

The *NumberOfFields* parameter refers to the number of fields that must be extracted from the data that are read from each line based on the NS-2 manual. In order to extract these data fields some modifications must take place on some of the elements contained in each line. These modifications are described by the metafile using special flags as will be explained. Furthermore, the *NumberOf-Fields* is used to ensure that the metafile's structure is correct. The value defined by the *NumberOf-Fields* parameter must be the same as the number of lines in the metafile that start with the *-name* flag since these lines are the ones who describe the trace file's structure. These lines contain information about the actual elements of a trace file of a specific format as described by the NS-2 manual and if their number is not equal to the *NumberOfFields* the structure of the metafile is considered corrupt and the metafile cannot be used.

The last element of the first three is the *UniqueCounter* flag, this flag is used to define the number of unique characters that must be matched in each line read from the trace file during the trace file recognition process. If a trace file's lines contain the number of unique characters that a metafile defines, then the trace file is matched with it and TRAFIL can start the actual trace file processing. The *UniqueCounter* though only defines the number of unique characters that must be matched; the actual unique characters are marked by the *-unique* flag as it shown in Figure 3.

Following the first three parameters are the fields which describe the columns of the trace file. For a normal trace file these lines are shown in the figure above. There are 14 different columns and the *-name* flag is used to define the name of the column, the *-type* flag is used to define the data type of this column and the *-index* flag is used to show its index inside the actual trace file. These three fields are mandatory for every line of the metafile that describes a trace file's column because these fields are also used by TRAFIL to create the table that will contain the trace file's data in the database. For this reason the data types that will be defined using the *-type* flag must conform to MySQL's supported data type syntax.

Besides these three flags there are also some other flags that serve specific purposes like declaring a unique sequence of characters that must be present in this column. For this purpose the *-unique* flag is used, this flag defines a character sequence that must be matched for all elements of a column for which it is set. Also there must be as many *-unique* flags as are defined using the *UniqueCounter* parameter. This way if all the unique character sequences are found they can be verified using the *UniqueCounter* parameter and the match can be established.

Another utility flag is the -*delimiter* flag, it is employed to signal a sequence of characters that will be used to divide a complex element into two other components. This is the case that was mentioned earlier in which the number of columns is different than the number of fields that must be extracted from the line. In these cases some elements are connected by a character sequence which is signalled by the -*delimiter* flag and that way TRAFIL can separate them.

Finally there are two other flags the -*startsWith* and -*endsWith*. These two as their name suggests are used to define some character sequences that elements of any column for which they are set either start or end with. These character sequences are not useful and must be removed in order to retrieve the useful information, using these flags TRAFIL can remove these characters.

The remaining lines are the "metric fields". These fields are used to denote the standard metrics the tool produces for each trace file. Each flag states the metric itself and is followed by the columns that will be used to produce that specific metric. The column names are recognized by the *–column* flag that precedes them. In each metafile all these metric flags must always be present, if they are not, the metafile is considered to be corrupt and the metric production phase cannot be completed. It is obvious that for different kinds of trace files the number and type of columns used to extract each metric might be different.

4.2 Sub-Metafiles

As mentioned in the previous sub section in order to process each trace file TRAFIL introduces the idea of using metafiles to describe the format of the input. It is often though necessary for a metafile to have some other utility files that can be used in situations where a metafile alone is not adequate. In the trace file processing procedure these situations occur when a trace file may include an arbitrary number of different header fields in each line and therefore have a lot of alternative forms.

The structure of a trace file depends greatly on the simulation scenario. Even in its own content a trace file may contain lines that are different with each other in the number of elements they contain. This is usually the case when the scenario involves traffic with different packet types travelling along the simulated network and using different routing protocols.

That is why TRAFIL uses a number of sub metafiles to complement the use of metafiles. Each trace file can log a number of different header fields, so a very straightforward way to handle all the different patterns is also to enable each metafile to have a number of different sub metafiles.

Actually for every one of the three different trace file formats that NS-2 produces TRAFIL has a different metafile and for each metafile there is a number of sub metafiles that is the same as the number of different header fields a trace file of a specific format can contain.

The structure of the sub metafiles which are used along with the metafile that encodes the structure of a Normal trace file is shown below:

```
NumberOfFields 4
NumberOfColumns 4
UniqueCounter 4
-name SourceLatitude -type double -index 1 -unique .
-name SourceLongitude -type double -index 2 -unique .
-name DestinationLatitude -type double -index 3 -unique .
-name DestinationLongtitude -type double -index 4 -unique .
```

Figure 4 Satellite Sub Metafile

```
NumberOfFields 4
NumberOfColumns 4
UniqueCounter 1
-name AckNumber -type int -index 1
-name FlagsTCP -type varchar(7) -index 2 -unique 0x
-name HeaderLength -type int -index 3
-name SocketAddressLength -type int -index 4
```

Figure 5 TCP Sub Metafile

Figure 4 depicts the structure of the sub metafile used to represent a satellite packet's header information which is logged in a Normal trace file. Figure 5 shows the structure of a sub metafile that represents a TCP packet's header information. The structure follows exactly the same conventions as were described earlier for a metafile. The same flags are used as in a metafile and the same first 3 mandatory fields must always be present as in a metafile. The sub metafiles shown above are only used after the trace file is matched with the Normal metafile. They are not used in the trace file recognition process; they are used after this process to enable the correct transfer of the trace file to the local database. They are also used when a user wishes to load a trace file from the database that was matched with the normal metafile.

5. GRAPHICAL SIMULATION SCENARIO SETUP

TRAFIL has taken successive steps towards automating the NS-2 simulation experience. An advanced step of that process was simplifying and automating most of the scenario creation process.

All simulations in NS-2 are presented in an OTcl script that describes the topology and the simulation parameters. This has to be prepared beforehand and requires knowledge of OTcl language (or even C++) from the user, while also being familiar to typical and advanced NS-2 objects and their parameters. Therefore, we introduced a new functionality to TRAFIL that enables the user to design the simulation using a graphical interface.

This interface is designed in such way that it is easy to learn and allows for quick scenario setup. Through the GUI users can pick network components and place them inside the simulation topology panel. These components can be typical network objects such as wired or wireless nodes and links that are formed between them. Each of them can be selected from a palette of components next to the simulation topology panel, and after its placement it can be customized via a pop-up menu that contains any available parameters for that object. All these network components are identical to the objects supported by NS-2, therefore after the user has finished designing the topology, it is translated to an OTcl script which can be executed by NS-2.

This new module is not separate from the rest of TRAFIL functions. Rather, it is integrated in a way that it synergizes with the rest functionalities. After describing the topology, users can generate the OTcl script and, if NS-2 is available in the current environment, proceed to the simulation execution and post-simulation analysis using TRAFIL tools only. This gives the whole process better transparency, since it omits the underlying procedures, jumping from scenario design to execution, and ultimately to results analysis. Given TRAFIL's performance in fast trace file analysis, this integration also allows for quick changes in the scenario, which in a sense make it easier for users to follow a trial-and-error method until they reach the desired scenario results.

5.1 Simulation Design Plane

The simulation design feature enables the user to describe a network in a way that is closer to designing rather than programming. Figure 6 describes its architecture which is organized in 4 layers.

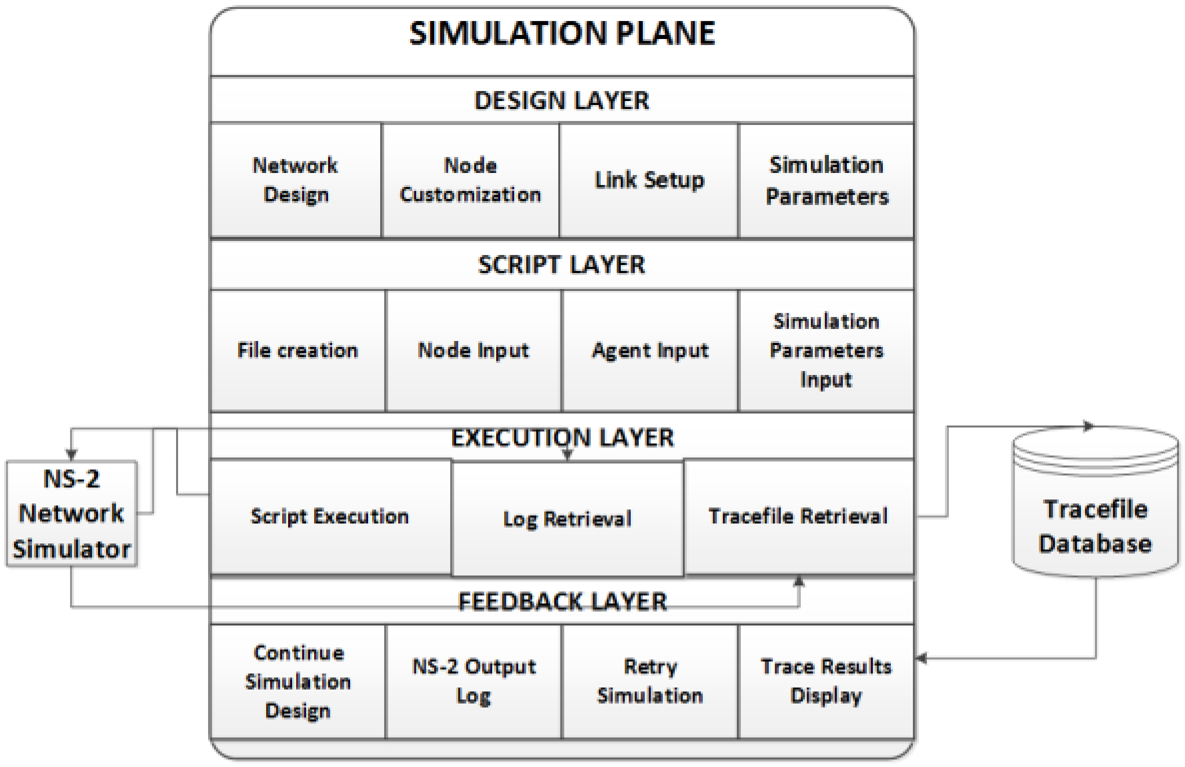


Figure 6 Simulation plane architecture

The first (and most important) layer is the design layer. A key part of this is the simulation topology panel or simulation design plane; a design environment similar in terms of layout to most modern design programs. It consists of the component palette, the design panel and several pop-up menus.

The palette includes all of the input options for the design panel, such as node types, links, lists of connected objects and a few buttons that are responsible for other scenario parameters. The rest of the simulation design plane is filled with the design panel, where the network topology schematic representation will be displayed. Depending on the network object type selected from the palette, an according shape will be painted there, resembling a network component. That shape also provides access to the menu related to its individual component.

All menus appear in separate windows, and allow for node customization, connecting existing wired nodes using links or specifying parameters necessary for the scenario that are not part of the visual design. These parameters include simulation scheduling, output file names etc. Users can keep the menu windows open while still accessing the design panel to edit or add more features to the network.

The next layer is the script layer. It transforms the network topology as shown, and its underlying data, and creates a simulation script file appropriate for NS-2, using the OTcl language's rules. The script file generation follows a specific format native to TRAFIL, which allows for a network layout reconstruction in the design panel, for later use. This allows users to save and load again later their work, or manipulate scripts that were previously created by TRAFIL without having to rearrange components in the topology layout. Nonetheless, the specific format does not mean a specific file type too. The output files are still normal tcl file types.

The next layer is the execution layer. This is an optional layer; depending on the user's operational system as well as whether NS-2 is installed, the user can immediately test the new scenario. The script generated in the previous layer is 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 parsed and saved in TRAFIL's database, producing metrics along the way.

The last layer is the feedback layer. Here TRAFIL displays a menu with the option to display the data or the output log produced by the simulation. Moreover, there are options to rerun the simulation, or return to the design layer and continue where the user left off.

5.2 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 of 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 using 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 anyway inside the design panel.

TRAFIL's new module gives users the opportunity to create a whole network pattern of nodes, and assign them any available options, using a pop-up menu designed for easy node parameter configuration. In order to simplify the process, we avoided manual entry of values as much as possible, using drop-down menus and pre-set values. This way the user can maintain control of the topology without having to deal with OTcl node, or even NS-2 documentation, to find out what his options are. All NS-2 available node parameters can be edited and correspond to the ones that are set using OTcl. The difference lies in the fact that the user is not obligated to have any previous knowledge of the actual commands to set them. In order to edit the parameters of a specific node the user can simply double-click or right-click a node in the design panel, revealing the parameter edit pop-up window.

5.3 Agents - Traffic Generators/Applications

In the previous sub section we explained how to create and configure network nodes. Although nodes and their proper setup are a major part of a simulation topology, they are not enough to power up a scenario, since they need a way to send packets to each other. For this purpose each node has to be associated with an agent, and possibly with a traffic source as well.

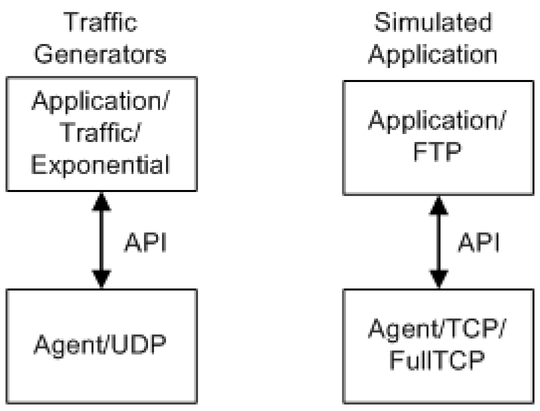


Figure 7 Connection between agent and traffic source in NS-2

Figure 7 illustrates how an agent and a traffic source are connected. NS-2 offers a variety of agents (sending or receiving) as well as traffic sources. Traffic sources are divided in two major categories: traffic generators (like the one used in Figure 7) and already simulated applications. Traffic sources can be attached to almost any type of agent. This is not the case though for simulated applications, which send their packets through a TCP transport agent only.

TRAFIL uses each node's configuration menu to assign agents and traffic sources to that node. Currently it provides support to all current NS-2 traffic sources, and the most common agents, which are TCP, User Datagram Protocol (UDP) and Null. The support integration is designed in such way so that when a user selects a type of agent for a protocol, TRAFIL shows the appropriate applications. Similarly, selecting an application shows you all of its specific NS-2 parameters.

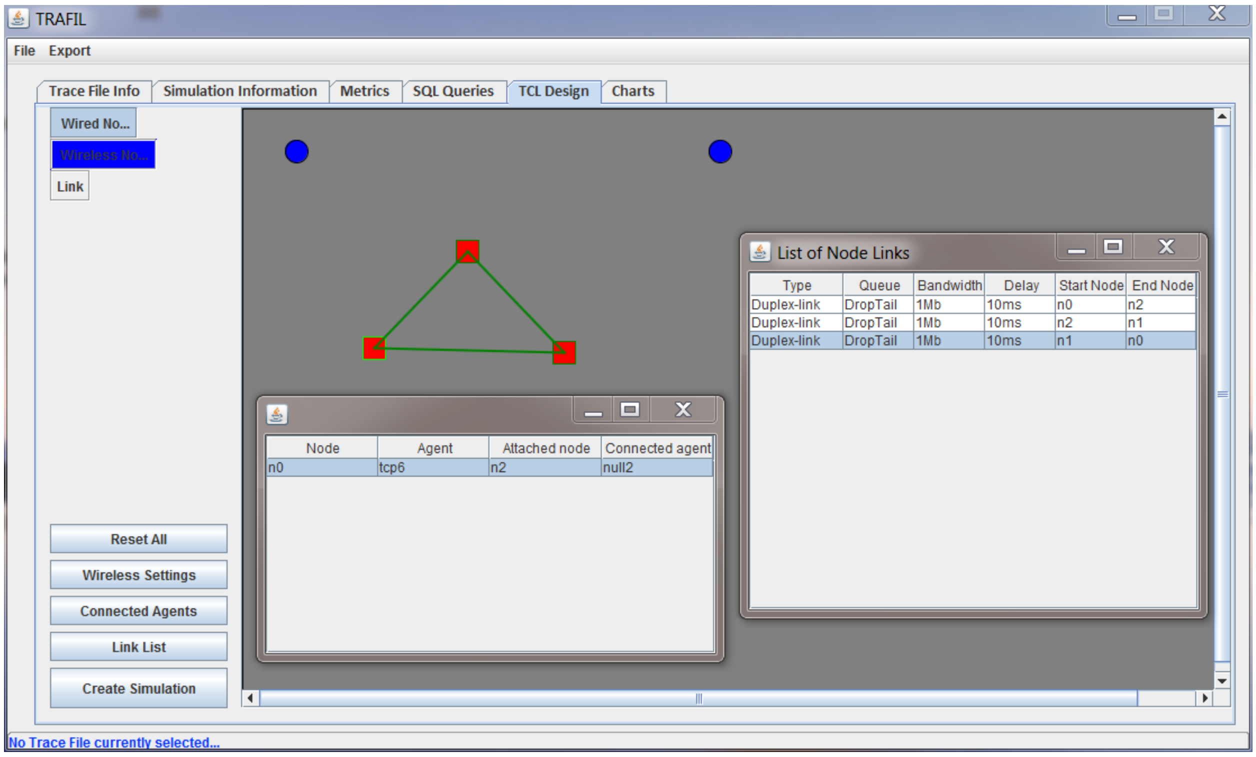
5.4 Links

However, creating and configuring nodes alone is not always enough to define a topology. Some simulation scenarios use wireless nodes, while others define wired networks. In the latter case, links have to be defined between nodes. This can be done using topology panel, and all link details are shown in a table window that displays existing links. Specific link parameters, such as bandwidth, delay, and queue type are configured using this window. There is a variety of link types available in NS-2 which are supported by TRAFIL.

In order to establish a link between two wired-type nodes a user has to select a new link from the palette, and then click on the two nodes to be linked. The new link is a shown as a straight line between the nodes, and its parameters can then be edited in the window described above.

5.5. Simulation design example

Figure 8 depicts the Graphic User Interface of TRAFIL's design module. In the example shown, 3 wired nodes (squares) are set up and linked together. There is also a wireless node present (circle). Both links are listed in the open link window, showing their editable parameters.



Type	Queue	Bandwidth	Delay	Start Node	End Node
Duplex-link	DropTail	1Mb	10ms	n0	n2
Duplex-link	DropTail	1Mb	10ms	n2	n1
Duplex-link	DropTail	1Mb	10ms	n1	n0

Node	Agent	Attached node	Connected agent
n0	tcp6	n2	null2

No Trace File currently selected...

Figure 8 TRAFIL simulation design

6. TRACE FILE ANALYSIS

This section describes one of TRAFIL’s core procedures and functionalities. Before any analysis on the results of a simulation, a trace file is firstly classified in terms of type and properly processed in order for its data to be retrieved and stored. Following these operations, users can then retrieve measurements and plot charts based on the trace file data.

6.1 Identification, Processing and Storage

In order for a meaningful analysis of a trace file’s data to take place, the trace file in question has to be classified. A trace file can be either one of NS-2’s supported formats (Normal, Old Wireless or New Wireless) or a user specified trace file format. The identification of a trace file is essentially a procedure of matching it with the appropriate metafile. In other words, TRAFIL identifies the meta-file whose encoding describes the trace file’s structure. In the case that no such metafile exists, then either the trace file structure is wrong or the metafile repository does not contain the appropriate metafile and the user should create one.

Having identified a metafile that properly describes the trace file in question TRAFIL proceeds to parse, process and finally store its data to the local database. Parsing and processing requires the use of all sub-metafiles that belong to the correct metafile since a trace file contains a variety of lines de-pending on the packet type or the events.

Algorithm 1.1: DATATRANSFER($metafile$)

```
while $newline \neq \{\}$
  do
    if LENGTH($newline$) $EQUALS\ metafile's\ NumberOfColumns\ value$
      then
        if $newline's\ Unique\ Character\ EQUALS\ metafile's\ Unique\ Characters$
          then
            $lineData \leftarrow$ PROCESS($newline$)
            STORE($lineData$)
    else if $newline's\ length > metafile's\ NumberOfColumns\ value$
      then
        $columnLength \leftarrow metafile's\ NumberOfColumns\ value$
        $lineData \leftarrow$ PROCESS($newline$)
        $residueData \leftarrow newline - lineData$
        while LENGTH($residueData$) $> 0$
          do
            for each $subMetafile \in\ Sub\ Metafile\ Repository$
              do
                if LENGTH($residueData$) $>= subMetafile's\ NumberOfColumns\ value$
                  then
                    if $residueData's\ Unique\ Characters\ EQUALS\ subMetafile's\ Unique\ Characters$
                      then
                        $lineData \leftarrow lineData +$ PROCESS($residueData$)
                        UPDATE($residueData$)
                        UPDATE($SubMetaFileList$)
            if LENGTH($residueData$) $NOT\ changed$
              then output ($error, New\ Line\ does\ not\ match\ any\ combination$)
    else
      output ($error, New\ Line\ does\ not\ match\ metafile\ structure.$)
```

Figure 9 Trace file processing algorithm

Using the metafile and sub metafile data the trace file is processed line by line. Figure 9 presents the pseudo code for the trace file processing operation which is one of the most important of TRAFIL's procedures since without properly and efficiently retrieving the data no further analysis can be made.

As shown in Figure 9, each line's number of elements is the first condition checked. That number is firstly checked if it matches that of the fields of the metafile and if that condition holds then the metafile is used to process the line in question. In addition to the number of elements criterion, an additional check is made to establish that indeed the line is described by the metafile using some "Unique Characters" that are defined in the metafile as it is also shown in Figure 9.

If the number of elements in a line is greater than the number specified by the metafile the sub metafiles are used. TRAFIL attempts to identify a combination of sub metafiles that can be used to process the additional fields. Thus, TRAFIL matches the additional fields with sub metafiles until no fields are left that have not been processed. If there is a subset of additional fields that cannot be matched to any sub metafile then the line is flagged as erroneous.

The identification and processing phase terminates when all the trace file lines have been processed. The next step is the data transfer to TRAFIL's local database and their storage. A new table is created and is named after the trace file. Its structure is based on the metafile that was used along with all its sub metafiles.

6.2 Statistics Calculation

Having correctly processed and stored a trace file or loaded a pre-existing one, a variety of information can then be retrieved. TRAFIL automatically calculates and presents the trace file's general simulation information without any user involvement. This information refers to the simulation as a whole and also to each specific simulation node. However, the user should specify the node for which he wants to view the general simulation information. More specific QoS metrics are calculated when the user requests them since such metrics refer to the communication between two specific nodes. The remainder of this sub section describes the exact general simulation information that can be retrieved using TRAFIL.

6.2.1 General Simulation Information

The general simulation information considers the simulation as a whole and can be viewed via TRAFIL's Simulation Information tab. It includes information such as the simulation's start time, end time and overall duration time. General simulation information about the simulation's nodes includes the Number of Nodes and Number of Sending Nodes. Moreover, the general simulation information includes communication information in terms of packets and bytes. More precisely, information such as: Number of Sent Packets, Number of Received Packets, Number of Dropped Packets, Number of Generated Packets and Number of Forwarded Packets. For each one of the aforementioned values there is a corresponding one calculated in bytes.

In the case of trace files that refer to wireless scenario simulations TRAFIL presents the following additional information: Number of Generated Packets and Bytes as well as Number of Received Packets and Bytes for each of the AGT (application layer), RTR (routing layer) and MAC (medium access layer) trace levels that NS-2 supports. Thus, users who are interested only in the packets and bytes regarding a specific communication layer can specify that layer and retrieve the information they want.

6.2.2 Node Specific Information

TRAFIL can also calculate similar information to the General Simulation Information, but for specific nodes again via the Simulation Information tab. Users can specify the node of interest and TRAFIL will calculate the general information that is specific to that node. This kind of information includes statistics such as the Number of Sent, Received, Dropped, Forwarded and Generated Packets. Similar to the General Simulation Information the same information is also shown in terms of bytes for that node. Furthermore, for wireless scenario simulations TRAFIL provides additional metrics for the AGT, RTR and MAC trace levels as it does for the general simulation information mentioned in 6.2.1.

6.2.3 QoS Parameters

In addition to General Simulation Information, TRAFIL enables users to calculate more specific metrics via the Metrics Tab. These are metrics that refer to the communication between simulation nodes. Thus, users specify the node pair of interest and TRAFIL provides the metrics that were mentioned in section 3.1 (TRAFIL Architecture). We specify here that the calculation of the Delay Jitter related measurements is based on the RFC 3550 [16] for RTP packets. Furthermore, if the trace file refers to a wired simulation the user must select the layer for which the calculation will be conducted. The options in this case are Link Layer or Physical Layer. If the simulation refers to a wireless scenario then the user must select between the 3 trace levels: AGT, RTR and MAC. After all the appropriate parameters have been set, the measurements can then be calculated. If there is no communication between the specified nodes at the particular layer then all the metrics are set to zero.

6.3 Plotting Charts

Chart plotting takes place via TRAFIL's Charts tab. TRAFIL supports the following charts: Packet Delivery Rate in packets/sec, Throughput in bytes/sec, Delay Jitter and Packet Loss Ratio. These charts can be drawn either for a node pair or for a single node. In addition, users should also specify the sampling rate in seconds and the trace level. For wired simulations the appropriate levels are Link Layer and Physical Layer and for wireless scenarios the appropriate trace levels are AGT, MAC and AGT. If there is no communication in the specified layer then an empty chart is displayed.

6.4 User initiated SQL Queries

TRAFIL's database stores each trace file in its own table. Each table's structure is based on the metafile that was used to process it. Thus, a table's columns are as many as the metafile's fields and all its corresponding sub metafiles' fields. Having each trace file stored in the database allows for faster loading and on-demand processing. Although the General Simulation Information and QoS metrics produced by TRAFIL cover the most common statistics, there is the possibility that a user will be interested in a measurement that currently is not offered by TRAFIL. Therefore, TRAFIL allows SQL queries directly to the database to alleviate this issue.

7. SIMULATION EXECUTION

Although TRAFIL was originally created as a post-simulation front-end framework, many pre-simulation features have since been added. For all these features to work, a function that communicates with NS-2 was created, allowing the user to execute any OTcl script using TRAFIL's graphic user interface. The results of such simulations are automatically imported in the local trace file data-base, allowing direct and seamless access to the results. Of course, it is required that the operational system natively supports NS-2 (i.e. unix systems).

There are several circumstances where TRAFIL uses this function, which are listed and explained below.

7.1 Simulation design plane scenarios

Following the process described in section 5 (Graphical Simulation Scenario Setup), the user can use the generated script file immediately for simulation. TRAFIL's generated scripts use a special extra notation in the form of comments, so that a user can save and later load the work he has done. They also have a standard structure, which consists of four parts: Script parameters (such as file names etc), node list, link list (and their parameters), agent information, and the simulation schedule.

If the Tcl script file creation is successful, TRAFIL saves it 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 in TRAFIL's database. However, if the script creation and execution was not successful, e.g. any parameters were missing or if the network layout was incomplete, the output of NS-2 will be shown in a report window, informing the user about the errors.

7.2 Simulation execution

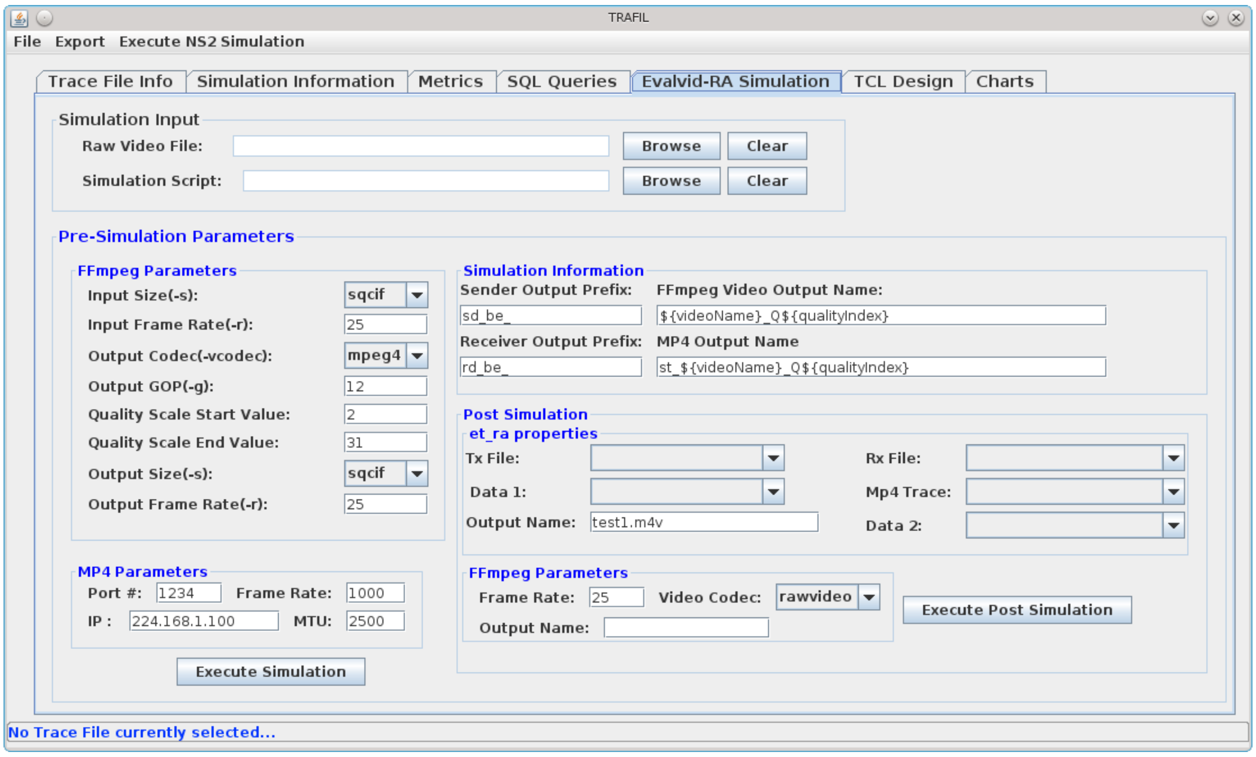
After a script has been created, TRAFIL sends it to NS-2 and retrieves the outcome, displaying it to the user accordingly. There is no direct interaction of the user with NS-2, since a successful simulation with proper outcome will automatically be parsed by TRAFIL and stored in its local trace file database.

Of course, it is possible that the user can input his own script file for simulation, bypassing the simulation design stage. This can be done using the top menu bar, where the user specifies the script file to be simulated, and starts the rest of the procedure right away, in a similar way to design plane scenarios.

7.3 Video simulation scenarios

One of the most popular NS-2 add-ons is Evalvid-RA, which enables video stream simulation across a network. Being an add-on, it also uses similar OTcl script files (adding a few extra parameters, such as video input file), making the simulation result and metrics extraction fairly similar as that of any other NS-2 simulation.

To that end, TRAFIL has also a special module that specializes in Evalvid-RA simulations and their specific parameters. Due to the complexity that an Evalvid-RA script file might have, the user has to input his own script file which describes the simulation topology and schedule. However, TRAFIL significantly simplifies the process of simulating such a scenario, by taking over all pre and post simulation stages, such as media file conversion and NS-2 simulation parameters.



Simulation Input
Raw Video File:
Simulation Script:

Pre-Simulation Parameters

FFmpeg Parameters
Input Size(-s): sqcif
Input Frame Rate(-r): 25
Output Codec(-vcodec): mpeg4
Output GOP(-g): 12
Quality Scale Start Value: 2
Quality Scale End Value: 31
Output Size(-s): sqcif
Output Frame Rate(-r): 25

MP4 Parameters
Port #: 1234 Frame Rate: 1000
IP : 224.168.1.100 MTU: 2500

Simulation Information
Sender Output Prefix: sd_be_
FFmpeg Video Output Name: ${videoName}_Q${qualityIndex}
Receiver Output Prefix: rd_be_
MP4 Output Name: st_${videoName}_Q${qualityIndex}

Post Simulation
et_ra properties
Tx File:
Rx File:
Data 1:
Mp4 Trace:
Output Name: test1.m4v
Data 2:

FFmpeg Parameters
Frame Rate: 25 Video Codec: rawvideo
Output Name:

No Trace File currently selected...

Figure 10 TRAFIL Evalvid-RA module

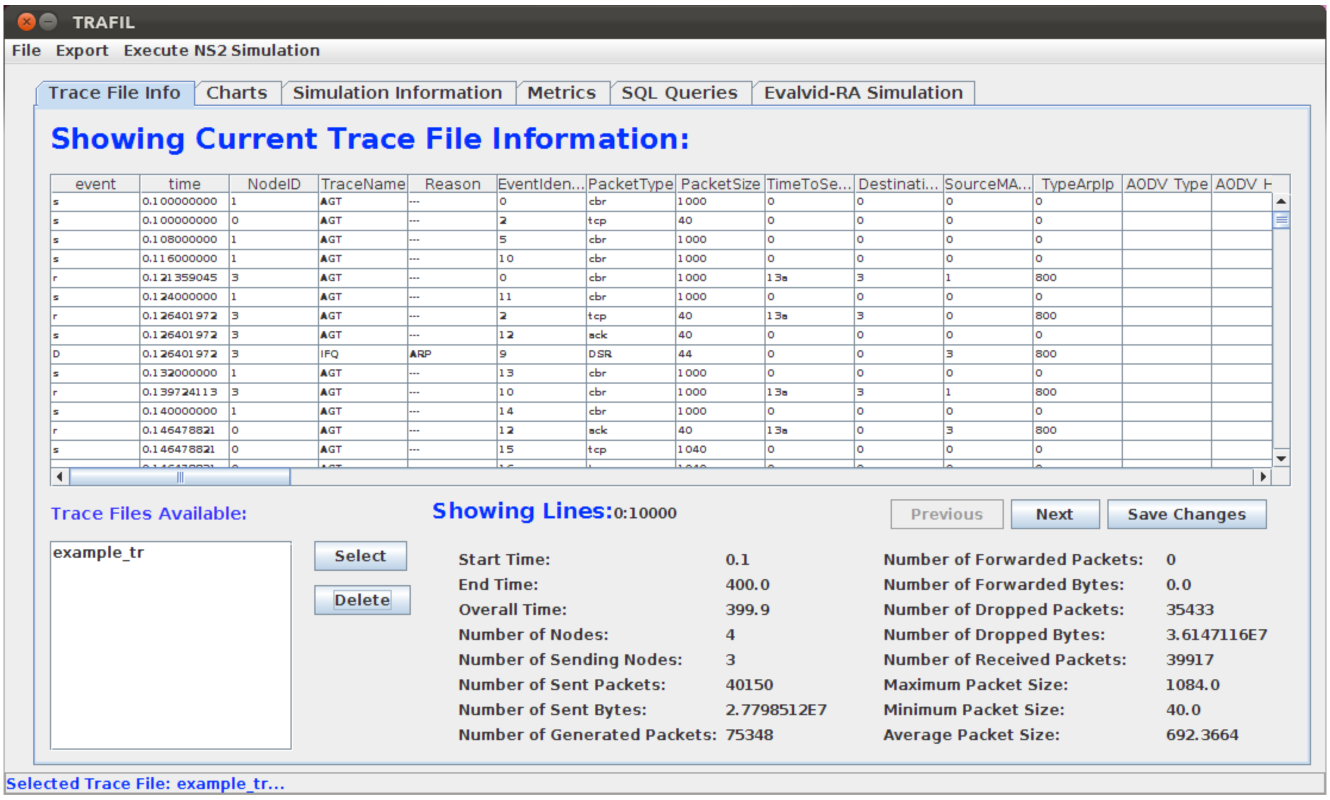
Figure 10 shows the Evalvid-RA module of TRAFIL and the parameters described above. As shown, the simulation procedure is split in two halves: the pre-simulation part (which includes the simulation itself) and the post-simulation. In the first part, the user sets the parameters for FFmpeg and MP4, as well as the video file and script path files. The input video file, which can be in any video format (*.yuv, *.mov, *.mp4 etc.), is converted to MPEG-4 (*.m4v format) using FFmpeg, and then gets prepared for simulation by the Evalvid MP4 tool. The preparation produces many different possible frame traces, which will be used by NS-2.

The second part of the procedure takes over the reconstruction of the video file and the re-encoding to a common video format file. The reconstruction is done using “et_ra”, a modified version of the original evalvid “et.exe” tool, which reads the produced packet Tx and Rx trace files as well as some dat files that contain information that assists in assembling the resulting MPEG-4 file (in m4v format again). Afterwards, FFmpeg is used to decode it back to its original format (presumably *.yuv). Then, PSNR is used to compare the decoded YUV file [17]. Finally, a report window shows the user the results of the simulation, as well as the produced files, which are typically stored in a default folder.

8. PLOTS AND RESULTS

In this section we present a usage example of TRAFIL in which we open a new trace file and show the information that the tool can extract. We present the calculation of the general simulation information, general node information, QoS measurements between nodes and charts that can be extracted using TRAFIL. The scenario that was used to create the sample trace file involved 4 wireless nodes of which 3 were stationary (nodes 0,1,2) and one mobile (node 3). The communication was between nodes 0-3 and 1-3 and the length of the simulation was 400 seconds. During this simulation node 3 moved every 50 seconds either closer or away from nodes 0 and 1 and finally at 300 seconds it moved as far as it could reach from nodes 0 and 1 inside the topology grid. This simulation is clearly a small one and it is only presented in order to introduce TRAFIL and portray its capabilities.

The resulting trace file was given as an input to TRAFIL and the procedures described in section 6 took place in order to identify the input's format. When the trace file has been analyzed and processed its contents are visible via the tool from a table as depicted in Figure 11. This is another feature that is unique in TRAFIL and its purpose is to enable users to have all the information they could possibly need centralized and ready to use. The table's columns are created based on the metafile and sub metafiles that were used to identify and process the trace file. In addition the general simulation information, consisting of simple statistics regarding the scenario, is also presented. The only non-trivial information is the Number of Generated Packets and its difference to the Number of Sent Packets. We consider as Number of Generated Packets every packet that was produced by a node and as sent packets the number of generated packets in every node minus the number of packets that were dropped in the same node they were created. The general information can also be viewed along with some extra fields regarding packets and bytes in each trace level of wireless scenarios in another part of TRAFIL named Simulation Information. The measurements in that area of the tool are created automatically after the trace file has been successfully transferred to the database. When a trace file that was the result of a wireless scenario is given as an input for analysis, for every trace level it contains information the corresponding extra fields are filled with the appropriate measurements. Thus, we calculate the same information for each trace level and that is the reason why in the Simulation Information part of TRAFIL are three fields for the Number of Generated Packets as well as for the Number of Generated Bytes.


TRAFIL

File Export Execute NS2 Simulation

Trace File Info | Charts | Simulation Information | Metrics | SQL Queries | Evalvid-RA Simulation

Showing Current Trace File Information:

event	time	NodeID	TraceName	Reason	EventIden...	PacketType	PacketSize	TimeToSe...	Destinati...	SourceMA...	TypeArpIp	AODV Type	AODV H
s	0.100000000	1	AGT	---	0	cbr	1000	0	0	0	0		
s	0.100000000	0	AGT	---	2	tcp	40	0	0	0	0		
s	0.108000000	1	AGT	---	5	cbr	1000	0	0	0	0		
s	0.116000000	1	AGT	---	10	cbr	1000	0	0	0	0		
r	0.121359045	3	AGT	---	0	cbr	1000	13a	3	1	800		
s	0.124000000	1	AGT	---	11	cbr	1000	0	0	0	0		
r	0.126401972	3	AGT	---	2	tcp	40	13a	3	0	800		
s	0.126401972	3	AGT	---	12	ack	40	0	0	0	0		
D	0.126401972	3	IFQ	ARP	9	DSR	44	0	0	3	800		
s	0.132000000	1	AGT	---	13	cbr	1000	0	0	0	0		
r	0.139724113	3	AGT	---	10	cbr	1000	13a	3	1	800		
s	0.140000000	1	AGT	---	14	cbr	1000	0	0	0	0		
r	0.146478821	0	AGT	---	12	ack	40	13a	0	3	800		
s	0.146478821	0	AGT	---	15	tcp	1040	0	0	0	0		

Trace Files Available: example_tr

Select Delete

Showing Lines:0:10000

Previous Next Save Changes

Start Time:	0.1	Number of Forwarded Packets:	0
End Time:	400.0	Number of Forwarded Bytes:	0.0
Overall Time:	399.9	Number of Dropped Packets:	35433
Number of Nodes:	4	Number of Dropped Bytes:	3.6147116E7
Number of Sending Nodes:	3	Number of Received Packets:	39917
Number of Sent Packets:	40150	Maximum Packet Size:	1084.0
Number of Sent Bytes:	2.7798512E7	Minimum Packet Size:	40.0
Number of Generated Packets:	75348	Average Packet Size:	692.3664

Selected Trace File: example_tr...

Figure 11 TRAFIL Main View

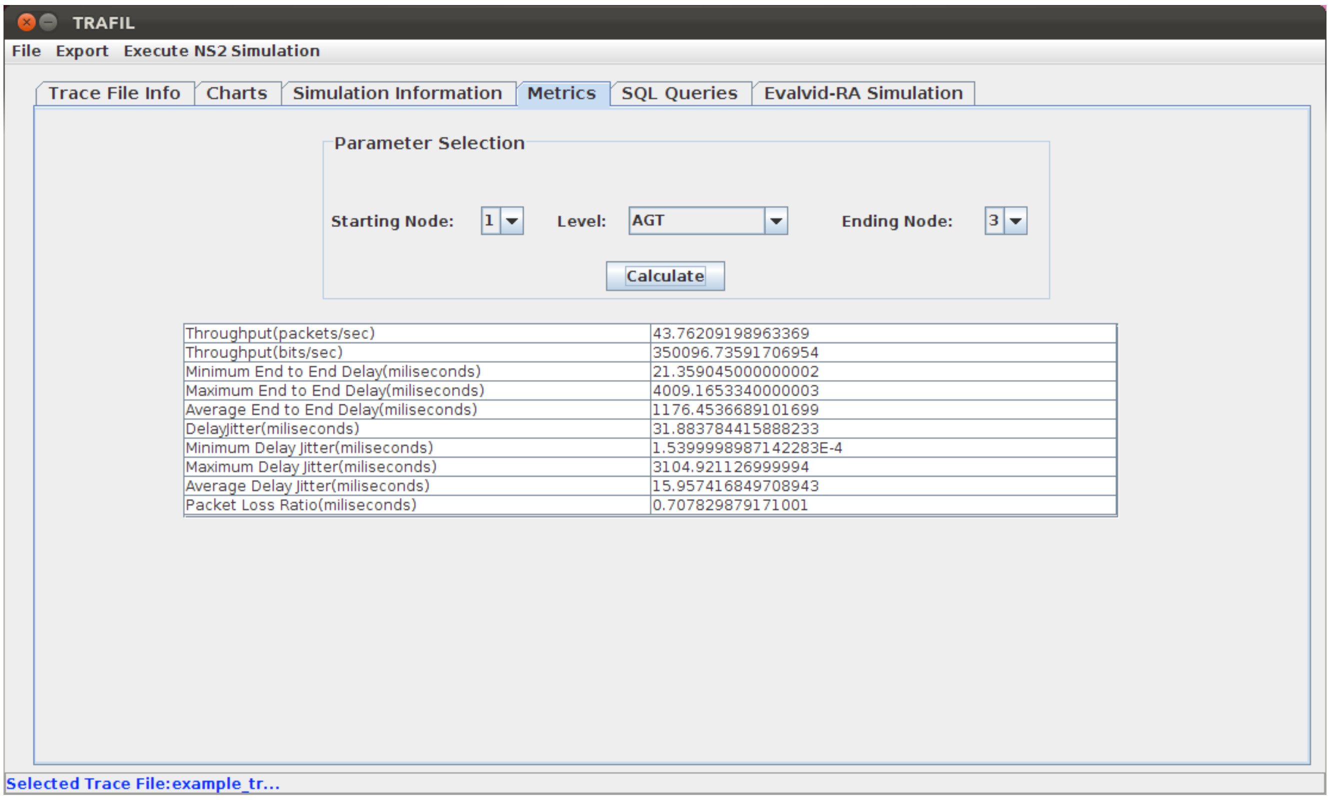
8.1 QoS Simulation Parameters Extraction

Once a trace file has been added to TRAFIL or loaded from the database it can be used to extract various QoS parameters such as Throughput, End to End Delay, Jitter and Packet Loss Ratio. These parameters are some of the most commonly calculated after a simulation and at the same time the most useful and informative in order to evaluate the performance of a network simulation. The results for our experiment are shown in Figure 12.

Packet delivery rate is calculated by dividing the number of packets that were successfully sent and received between the selected nodes at the time of arrival of the first and last packet between the nodes. To calculate throughput we divide the number of bits that were successfully sent and received between the selected nodes at the time of arrival of the first and last bit between the nodes. Each value is calculated in the space defined by the sampling rate.

The End to End Delay for each packet between the two nodes is calculated by finding the difference in the arrival time and transmission time of every packet that was exchanged from the sender node to the receiver node.

The Jitter related metrics as we have mentioned earlier are calculated based on the RFC 3550. Finally the Packet Loss Ratio is calculated by finding all the packets that were sent and received between the two nodes, dividing their difference by the packets sent and multiplying by 100.



Starting Node: 1   Level: AGT   Ending Node: 3

Throughput(packets/sec)	43.76209198963369
Throughput(bits/sec)	350096.73591706954
Minimum End to End Delay(miliseconds)	21.359045000000002
Maximum End to End Delay(miliseconds)	4009.1653340000003
Average End to End Delay(miliseconds)	1176.4536689101699
DelayJitter(miliseconds)	31.883784415888233
Minimum Delay Jitter(miliseconds)	1.5399998987142283E-4
Maximum Delay Jitter(miliseconds)	3104.921126999994
Average Delay Jitter(miliseconds)	15.957416849708943
Packet Loss Ratio(miliseconds)	0.707829879171001

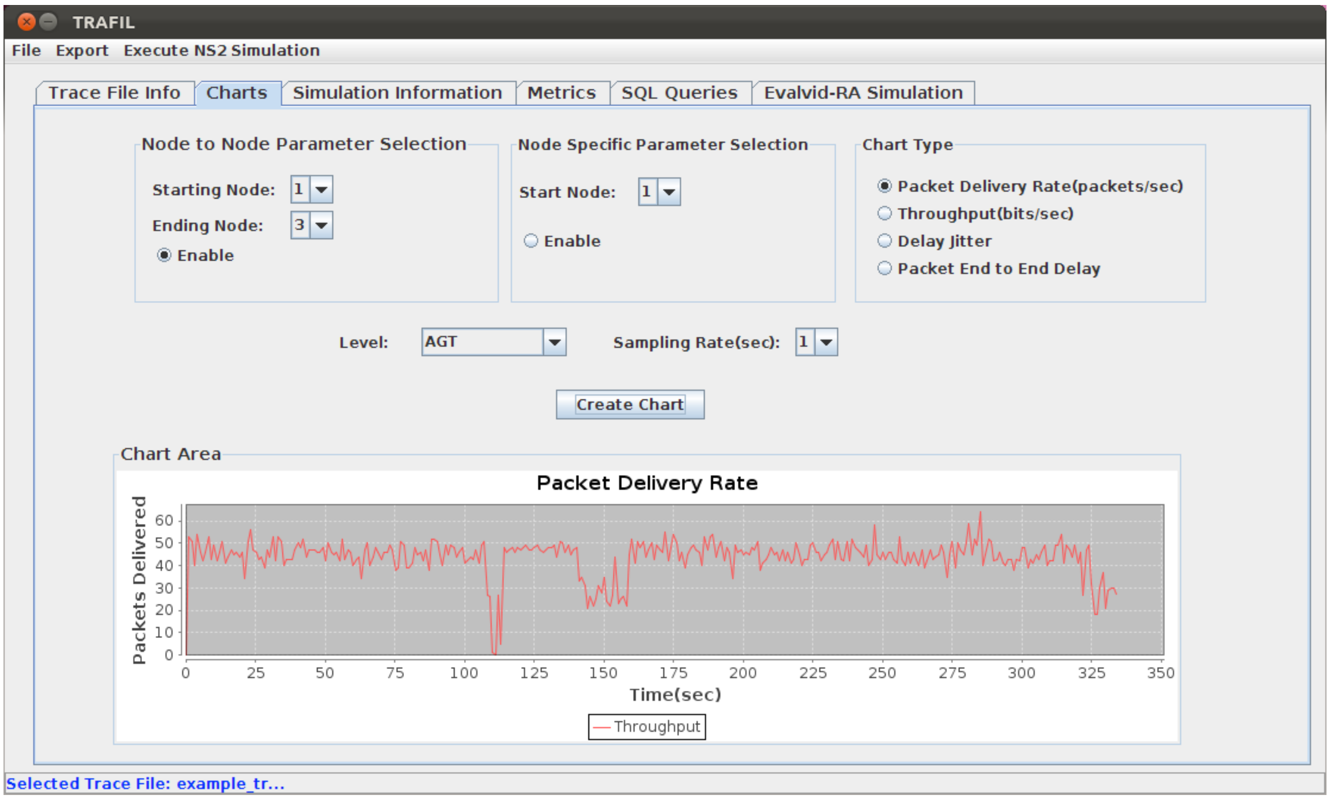
Figure 12 QoS Parameter Extraction

8.2 Chart Plotting

A useful utility offered by TRAFIL is to plot various charts based on the information contained in a trace file as shown in Figure 13 and Figure 14. The charts can refer either to a pair of nodes (Figure 13) or to a specific node (Figure 14). There are four types of charts that can be extracted using TRAFIL which are Packet Delivery Rate, Throughput, Delay Jitter and Packet Loss Ratio. Furthermore, the calculation in each case can be made in two distinct sampling rates: 1 and 5 seconds. The sampling rate defines the time interval in which we calculate the value of the selected chart. Finally there is also the opportunity to define the communication layer for which the information will be collected. Namely, for wired scenarios that yield Normal trace files the communication levels are Link and Network Layer and for wireless scenarios the corresponding levels are MAC, RTR and AGT layer. Thus, the user can define various parameters regarding the chart and obtain a more accurate result.

8.3 User SQL Query Execution

TRAFIL has been designed in such a manner that the user will be able to conduct his post simulation analysis and retrieve results with the least possible work. Nevertheless, there is no way to predict and implement all the different functionalities that a user might require during the analysis of a trace file. Therefore, TRAFIL offers the ability to execute SQL queries directly to the database in order to retrieve information from trace files that is currently not offered by the tool. The only queries that are supported are select queries and the reason is to protect the database from user errors that might lead to corrupting the system. An example of executing a query to retrieve all received packets from a trace file is shown in Figure 15.



Node to Node Parameter Selection
Starting Node: 1
Ending Node: 3
[✓ Enable]

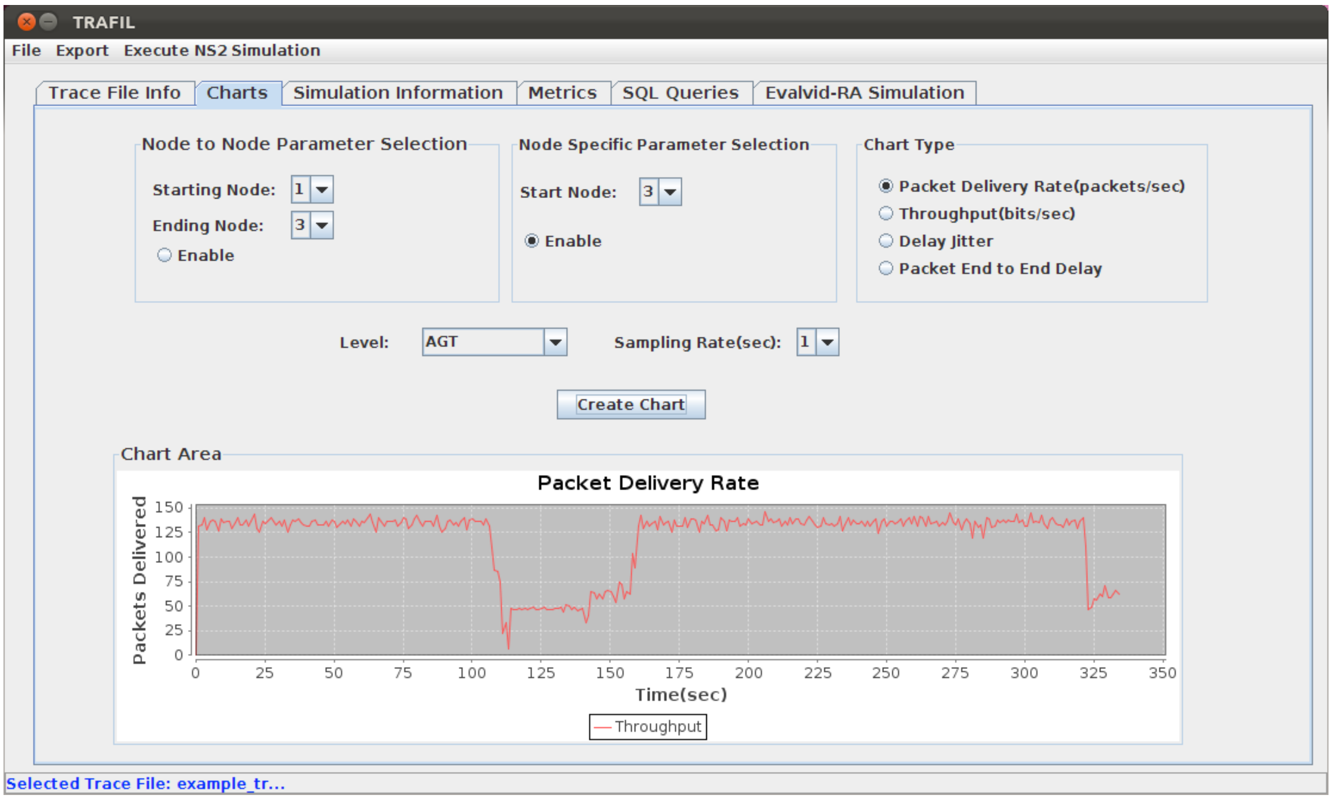
Node Specific Parameter Selection
Start Node: 1
Enable

Chart Type
[✓ Packet Delivery Rate(packets/sec)]
Throughput(bits/sec)
Delay Jitter
Packet End to End Delay

Level: AGT
Sampling Rate(sec): 1

Selected Trace File: example_tr...

Figure 13 Chart Plotting



Node to Node Parameter Selection

Starting Node: 1

Ending Node: 3

○ Enable

Node Specific Parameter Selection

Start Node: 3

● Enable

Chart Type

● Packet Delivery Rate(packets/sec)

○ Throughput(bits/sec)

○ Delay Jitter

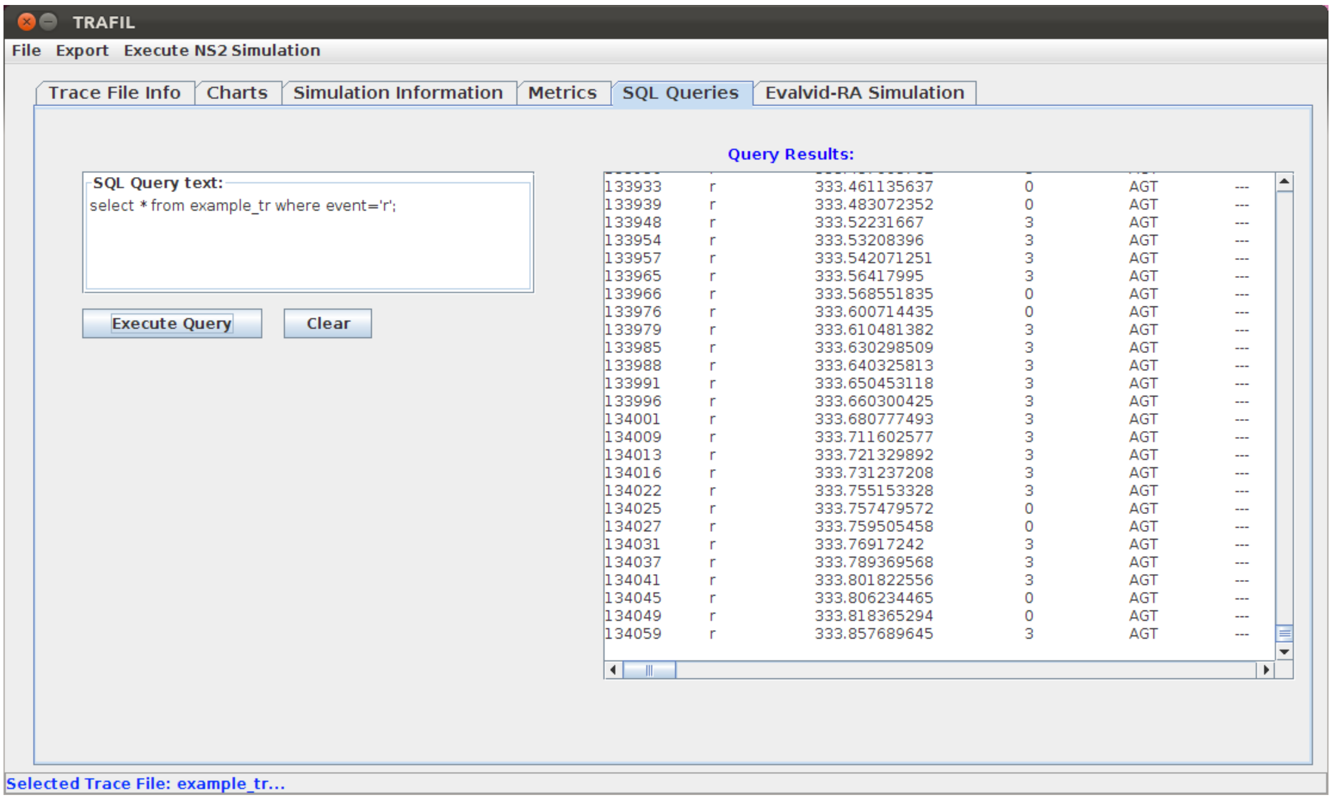
○ Packet End to End Delay

Level: AGT

Sampling Rate(sec): 1

Selected Trace File: example_tr...

Figure 14 Specific Node Chart



```
select * from example_tr where event='r';
```

```
133933   r   333.461135637   0   AGT   ---
133939   r   333.483072352   0   AGT   ---
133948   r   333.52231667    3   AGT   ---
133954   r   333.53208396    3   AGT   ---
133957   r   333.542071251   3   AGT   ---
133965   r   333.56417995    3   AGT   ---
133966   r   333.568551835   0   AGT   ---
133976   r   333.600714435   0   AGT   ---
133979   r   333.610481382   3   AGT   ---
133985   r   333.630298509   3   AGT   ---
133988   r   333.640325813   3   AGT   ---
133991   r   333.650453118   3   AGT   ---
133996   r   333.660300425   3   AGT   ---
134001   r   333.680777493   3   AGT   ---
134009   r   333.711602577   3   AGT   ---
134013   r   333.721329892   3   AGT   ---
134016   r   333.731237208   3   AGT   ---
134022   r   333.755153328   3   AGT   ---
134025   r   333.757479572   0   AGT   ---
134027   r   333.759505458   0   AGT   ---
134031   r   333.76917242    3   AGT   ---
134037   r   333.789369568   3   AGT   ---
134041   r   333.801822556   3   AGT   ---
134045   r   333.806234465   0   AGT   ---
134049   r   333.818365294   0   AGT   ---
134059   r   333.857689645   3   AGT   ---
```

Figure 15 SQL Query Execution

8.4 NS-2 via TRAFIL

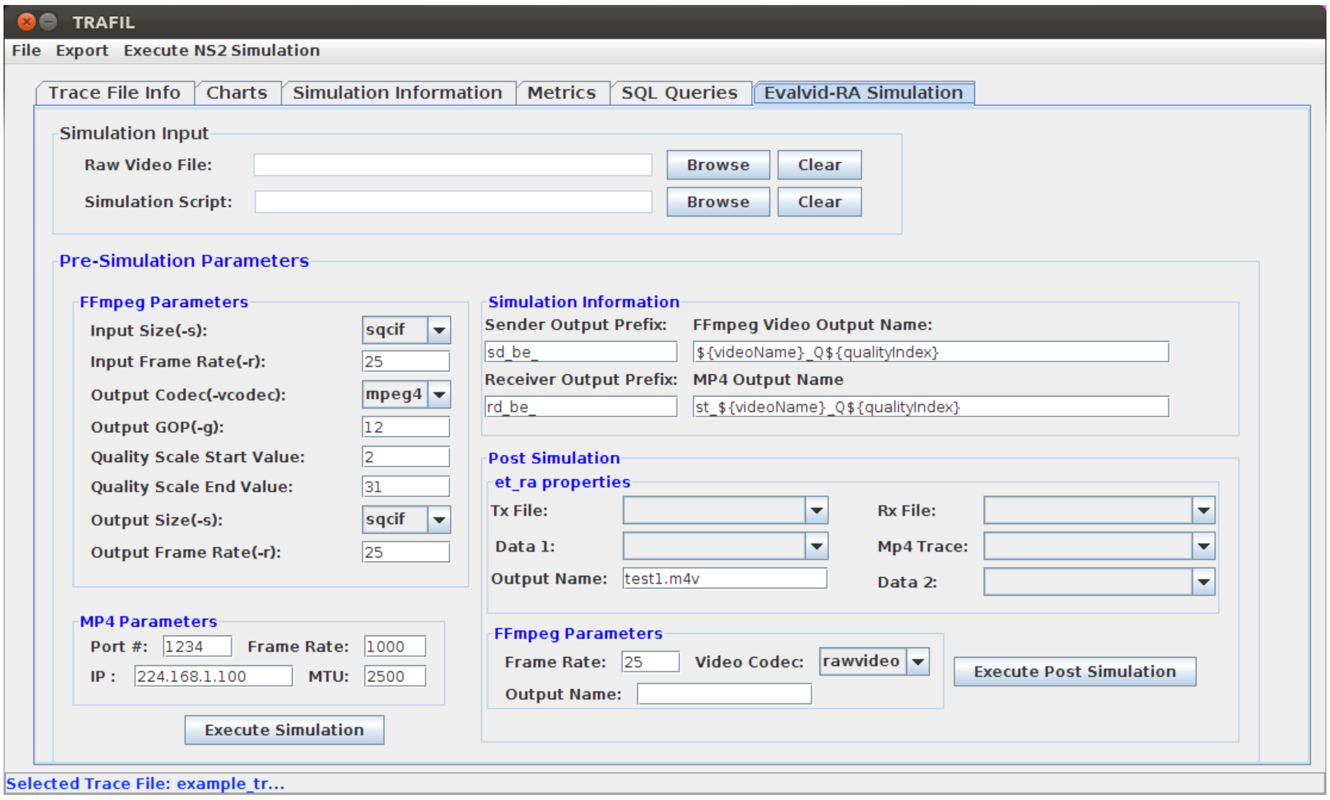
8.4.1 Executing OTcl Scripts

Every NS-2 simulation scenario is described and constructed using the OTcl scripting language. A user has the ability to create custom wired or wireless scenarios with an arbitrary number of nodes, protocols, application clients and traffic generators. Besides the commands that are used to create the simulation plane, the user can create a number of objects that can be used to control and monitor the actual simulation like monitor objects and random generators. Finally, after creating the OTcl script in order for it to be executed the user must invoke NS-2 giving the script as a parameter. After the simulation has concluded in the majority of cases the most important data reside in the created trace files. Thus, a user must find a way to process these trace files either using scripting languages like AWK and Perl or using post simulation analysis tools like TRAFIL, jTrana or Tracegraph.

Although these are the steps for executing a certain simulation scenario, TRAFIL enables the user to execute an OTcl script through the tool. TRAFIL will execute the specified simulation scenario, locate the resulting trace file and start the trace file analysis procedure described in section 6.

8.4.2 Simulating Video Transmission using Evalvid-RA

Simulating video transmission is one of the most common uses of NS-2 and it is usually implemented using the Evalvid-RA framework. Therefore, TRAFIL has automated this procedure as shown in Figure 16. The only requirement is that NS-2 is installed on the system and Evalvid-RA has been incorporated correctly. The simulation procedure is broken into two specific steps, the Pre Simulation and Post Simulation phases. In the Pre Simulation phase the raw video file as well as the simulation scenario must be specified. In addition, the Pre Simulation phase includes the raw video's processing using FFmpeg and its transmission using the MP4 tool. Thus, their parameters must be specified; TRAFIL has already set some default parameters which are the ones defined by Evalvid's own examples that accompany its source code. Finally, in order to conduct the Post Simulation phase the user must specify the names of the files he uses in his TCL script to read the video traces produced by MP4 and the names of the output receiver and sender files of his script. The file names must be specified because these files are crucial in the successful execution of the whole simulation procedure. The files are given as parameters at FFmpeg and et_ra and if they are not specified in advance there is no way for TRAFIL to complete the simulation without problems.



Simulation Input

Raw Video File:

Simulation Script:

Pre-Simulation Parameters

FFmpeg Parameters

Input Size(-s): sqcif

Input Frame Rate(-r): 25

Output Codec(-vcodec): mpeg4

Output GOP(-g): 12

Quality Scale Start Value: 2

Quality Scale End Value: 31

Output Size(-s): sqcif

Output Frame Rate(-r): 25

MP4 Parameters

Port #: 1234 Frame Rate: 1000

IP : 224.168.1.100 MTU: 2500

Simulation Information

Sender Output Prefix: sd_be_

FFmpeg Video Output Name: ${videoName}_Q${qualityIndex}

Receiver Output Prefix: rd_be_

MP4 Output Name: st_${videoName}_Q${qualityIndex}

Post Simulation

et_ra properties

Tx File:

Rx File:

Data 1:

Mp4 Trace:

Output Name: test1.m4v

Data 2:

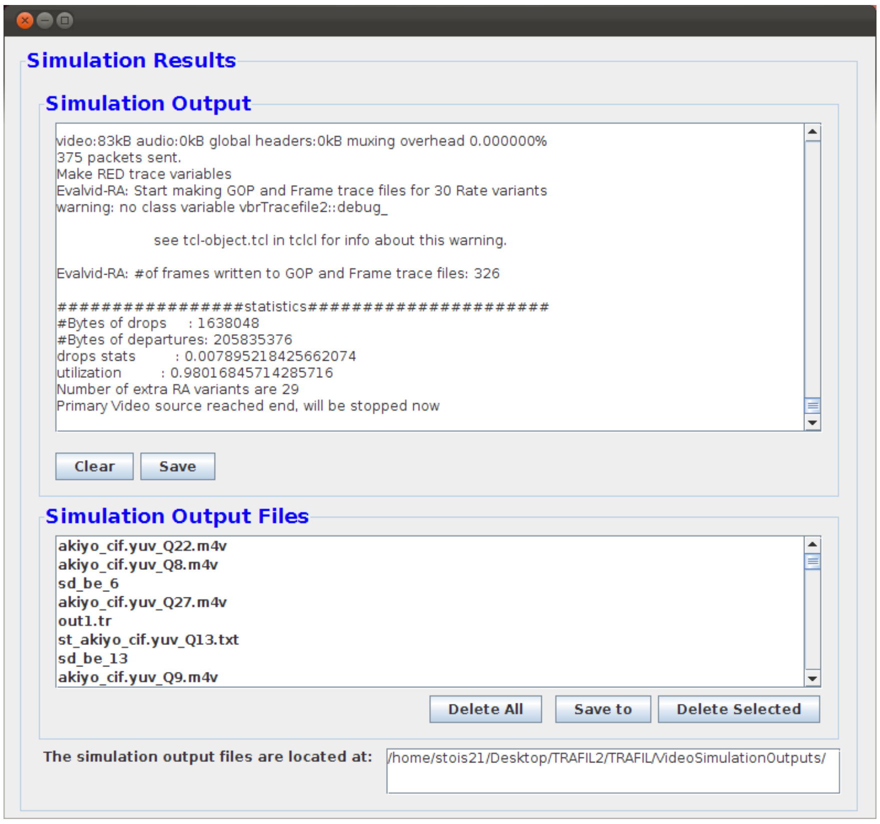
FFmpeg Parameters

Frame Rate: 25 Video Codec: rawvideo

Output Name:

Selected Trace File: example_tr...

Figure 16 Video Simulation

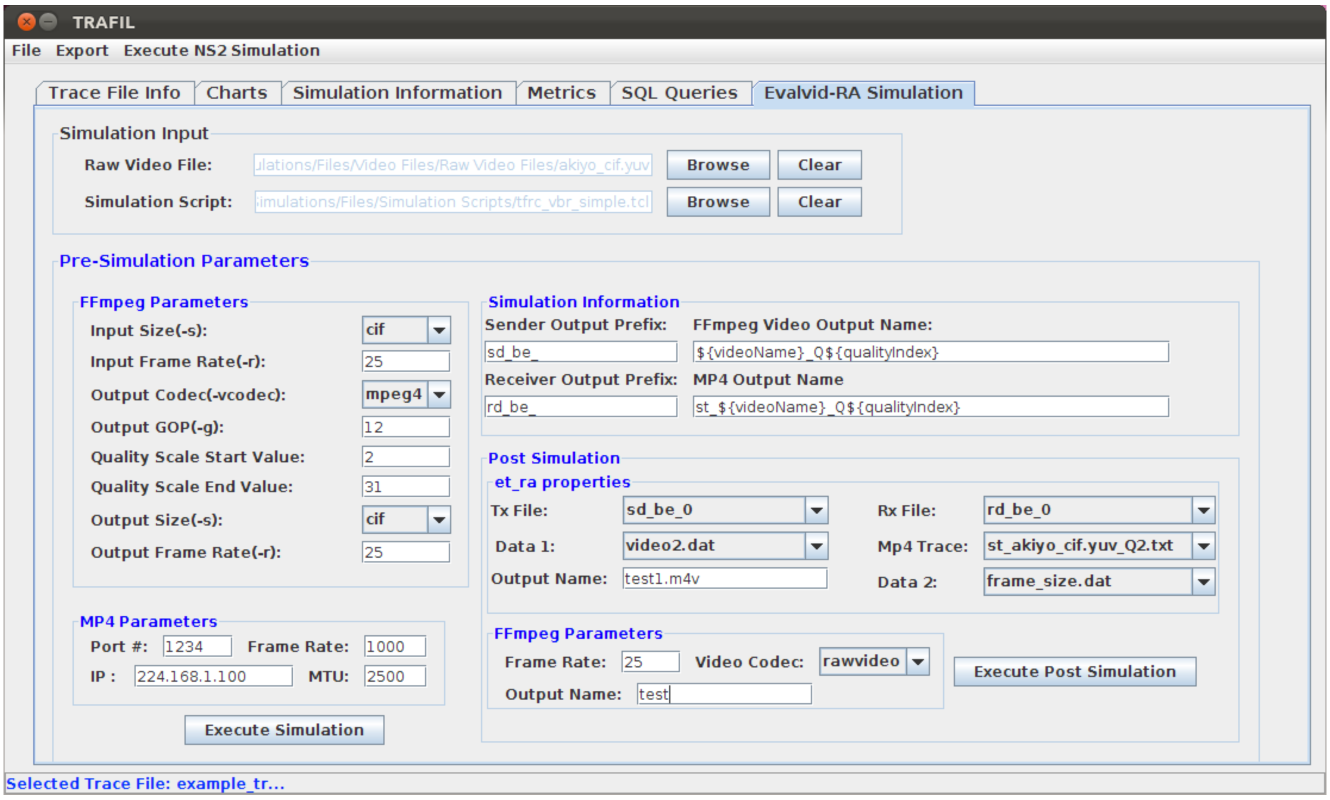


```
video:83kB audio:0kB global headers:0kB muxing overhead 0.000000%
375 packets sent.
Make RED trace variables
Evalvid-RA: Start making GOP and Frame trace files for 30 Rate variants
warning: no class variable vbrTracefile2::debug_

                 see tcl-object.tcl in tclcl for info about this warning.

Evalvid-RA: #of frames written to GOP and Frame trace files: 326

#################statistics######################
#Bytes of drops     : 1638048
#Bytes of departures: 205835376
drops stats         : 0.007895218425662074
utilization         : 0.98016845714285716
Number of extra RA variants are 29
Primary Video source reached end, will be stopped now
```

```
akiyo_cif.yuv_Q22.m4v
akiyo_cif.yuv_Q8.m4v
sd_be_6
akiyo_cif.yuv_Q27.m4v
out1.tr
st_akiyo_cif.yuv_Q13.txt
sd_be_13
akiyo_cif.yuv_Q9.m4v
```

The simulation output files are located at: /home/stois21/Desktop/TRAFIL2/TRAFIL/VideoSimulationOutputs/

Figure 17 Pre-Simulation phase results



Simulation Input

Raw Video File:	ulations/Files/Video Files/Raw Video Files/akiyo_cif.yuv
Simulation Script:	imulations/Files/Simulation Scripts/tfrc_vbr_simple.tcl

Pre-Simulation Parameters

FFmpeg Parameters

Input Size(-s):	cif
Input Frame Rate(-r):	25
Output Codec(-vcodec):	mpeg4
Output GOP(-g):	12
Quality Scale Start Value:	2
Quality Scale End Value:	31
Output Size(-s):	cif
Output Frame Rate(-r):	25

MP4 Parameters

Port #:	1234	Frame Rate:	1000
IP :	224.168.1.100	MTU:	2500

Simulation Information

Sender Output Prefix:	sd_be_
FFmpeg Video Output Name:	${videoName}_Q${qualityIndex}
Receiver Output Prefix:	rd_be_
MP4 Output Name	st_${videoName}_Q${qualityIndex}

Post Simulation

et_ra properties

Tx File:	sd_be_0	Rx File:	rd_be_0
Data 1:	video2.dat	Mp4 Trace:	st_akiyo_cif.yuv_Q2.txt
Output Name:	test1.m4v	Data 2:	frame_size.dat

FFmpeg Parameters

Frame Rate:	25	Video Codec:	rawvideo
Output Name:	test		

Selected Trace File: example_tr...

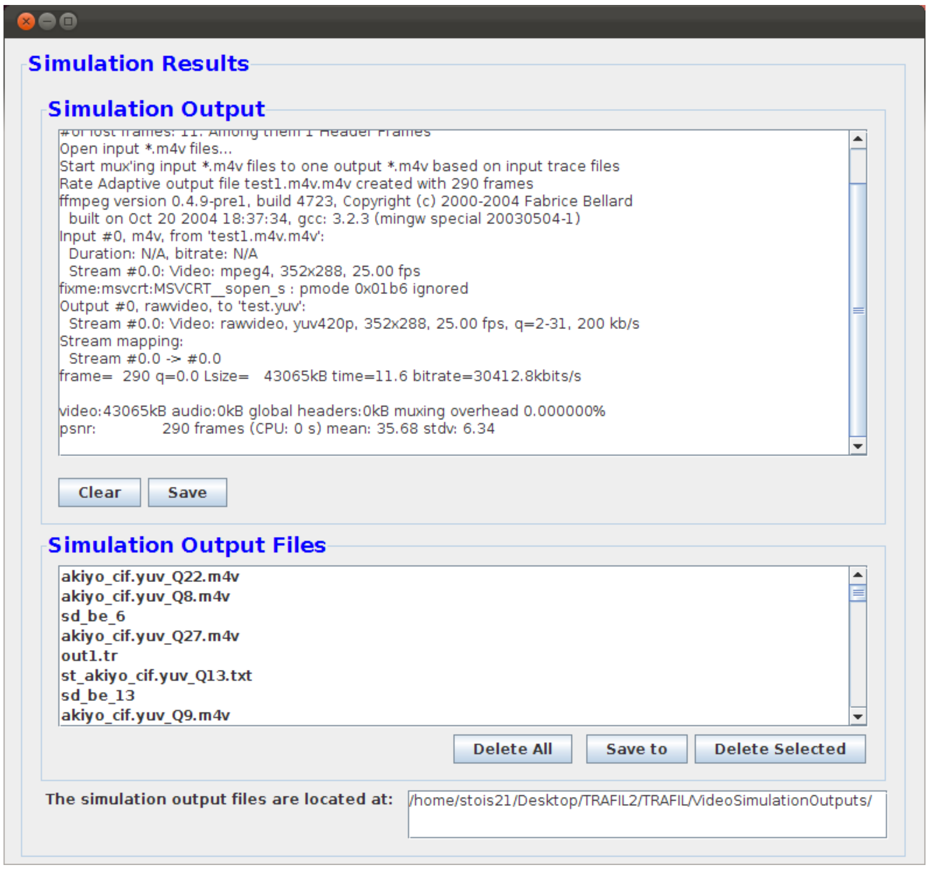
Figure 18 Post-Simulation phase

When all the required parameters have been set the simulation can be executed. If the simulation has been successful then the Post Simulation parameters become available, namely the sender and receiver files, the .dat files as well as the MP4 video traces. Thus, the user is able to select various combinations of these files and retrieve QoS statistics for his video transmission. The first phase of the video simulation, the Pre-Simulation phase, is shown above in Figure 16. In this phase all the necessary parameters for the video encoding using FFmpeg must be defined.

Then using the MP4 tool trace files for the video transmissions are created and finally the execution of the simulation takes place by invoking NS-2. The results of a sample video simulation are shown in Figure 17. The results include the actual console output of all the involved tools (FFmpeg, MP4, NS-2) as well as all the created files of the simulation. These files are available in a specific folder in TRAFIL's file hierarchy and its path is also given. In addition, the opportunity is given to save the files to any folder in the operating system or to delete them from TRAFIL without accessing the folder in which they are stored. After the Pre-Simulation phase has concluded all the parameters in order for the Post Simulation phase to be executed become available as shown in Figure 18. All the created files are now available to be selected and they include the Tx(sender) and Rx(receiver) files, the data files(Data1, Data2) and MP4's output trace files. These files are all input parameters for the et_ra tool and the final parameter is the output video name.

In addition to the et_ra inputs the user must specify FFmpeg's parameters in order to reconstruct the video. Again some default parameters are available by TRAFIL.

Figure 19 illustrates the results of the Post-Simulation phase including the PSNR calculation of the video transmission example simulation. All the resulting files including the console output are available to the user. The resulting files are included in the same directory as the previously created file during the Post-Simulation phase. The user is thus able to re-use them in order to conduct further process and extraction of additional QoS parameters.



```
#of lost frames: 11. Among them 1 Header Frames
Open input *.m4v files...
Start mux'ing input *.m4v files to one output *.m4v based on input trace files
Rate Adaptive output file test1.m4v.m4v created with 290 frames
ffmpeg version 0.4.9-pre1, build 4723, Copyright (c) 2000-2004 Fabrice Bellard
  built on Oct 20 2004 18:37:34, gcc: 3.2.3 (mingw special 20030504-1)
Input #0, m4v, from 'test1.m4v.m4v':
  Duration: N/A, bitrate: N/A
  Stream #0.0: Video: mpeg4, 352x288, 25.00 fps
fixme:msvcrt:MSVCRT__sopen_s : pmode 0x01b6 ignored
Output #0, rawvideo, to 'test.yuv':
  Stream #0.0: Video: rawvideo, yuv420p, 352x288, 25.00 fps, q=2-31, 200 kb/s
Stream mapping:
  Stream #0.0 -> #0.0
frame=  290 q=0.0 Lsize=   43065kB time=11.6 bitrate=30412.8kbits/s

video:43065kB audio:0kB global headers:0kB muxing overhead 0.000000%
psnr:           290 frames (CPU: 0 s) mean: 35.68 stdv: 6.34
```

```
akiyo_cif.yuv_Q22.m4v
akiyo_cif.yuv_Q8.m4v
sd_be_6
akiyo_cif.yuv_Q27.m4v
out1.tr
st_akiyo_cif.yuv_Q13.txt
sd_be_13
akiyo_cif.yuv_Q9.m4v
```

The simulation output files are located at: /home/stois21/Desktop/TRAFIL2/TRAFIL/VideoSimulationOutputs/

Figure 19 Post-Simulation results

8.5 TRAFIL Results

In this section we present timing results about the core procedures described in section 6 which include identifying a trace file's type, parsing, processing and finally transferring its content to the local database. We have argued that TRAFIL, by using metafiles and sub metafiles, not only accomplishes to cope with all different kinds of trace files, but it does so exceptionally quick. Therefore, we present in Table 1 results regarding the aforementioned procedures for a variety of trace file sizes.

Table 1 Processing time results

Trace File Size(MB)	Execution Time(ms)
0.005	13
0.126	42
0.402	217
1	378
2	998
3	1,569
7	3,558
11	5,186
16	4,887
58	25,351
118	51,087

Table 2 Trace file formats execution comparison

Trace file size(MB)	Wireless scenario trace file processing time (ms)	Wired scenario trace file processing time (ms)
2	1,967	1,456
6	2,499	2,215
13	4,973	3,635

24	8,699	5,613
35	13,095	8,406
47	17,816	11,767
62	20,906	16,048

Based on the above results it is apparent that TRAFIL manages even for very large trace files to keep their processing time extremely low. For trace files that have size up to 10 MB the execution time is very small and it does not increase significantly, although for trace file sizes that exceed 10 MB the processing time starts to grow following a near linear increase. Another observation that can be deducted from Table 1 is that the trace file with size 16MB has a smaller processing time than the 11MB one and the reason is the trace file format. The trace file with size 11MB was produced using a wireless scenario and the 16MB trace file resulted from a wired scenario. In order to examine why the trace file format affects the processing time we measured TRAFIL's execution time for trace files of the same size both for wireless and wired scenarios and the results are shown in Table 2.

From the above results we can draw the conclusion that certainly trace files that are produced from wired scenarios take less time to be processed by TRAFIL rather than ones which result from a wireless scenario simulation. The reason is that during the trace file processing and parsing phases the number of sub metafiles that are tested against each trace file line is smaller for Normal trace files (produced by wired scenarios) compared to the number that is tested for Wireless trace file formats. The metafile used to process Normal trace files has two utility sub metafiles and the metafiles used to process Old Wireless or New Wireless trace files have eight and nine sub metafiles respectively. Nevertheless, although there is a difference in the processing time it is not a large one and the most important conclusion is that for both cases TRAFIL manages to keep the processing time extremely low.

Finally we present a comparison between TRAFIL and other popular tools that are used for trace file analysis: Tracegraph2.02 and jTrana 1.0. These tools have been described in section 2, they are among the most known for processing simulation trace files but one of their main drawback is the amount of time they need to open a trace file [18]. As we have demonstrated above TRAFIL behaves exceptionally well at this task, therefore we present in Table 3 a detailed comparison between these tools and TRAFIL:

Table 3 Trace File Analysis Tools Comparison

Trace file size(MB)	**TRAFIL Processing Time (ms)**	**Trace-graph Proc-essing Time (ms)**	**jTrana Processing Time (ms)**
2	1,967	27,000	29,700
6	2,499	75,400	68,000
13	4,973	197,600	135,200
24	8,699	495,100	177,700
35	13,095	949,300	272,000
47	17,816	1,090,700	362,700
62	20,906	2,097,100	486,800

The trace files used to create the measurements referred to wireless simulations and were the same for all three tools. In order to retrieve the timings both for Tracegraph 2.02 and jTrana 1.0 we manually measured the time it took for each tool to open a trace file. Furthermore, to obtain more ac-curate results we made each measurement multiple times and calculated the mean value. As it can be easily deducted from the above table TRAFIL yields timings that are considerably smaller than the other two tools (ranging from 15 times smaller to 100 times smaller). This fact becomes more obvious if we consider that TRAFIL's largest processing time is for the trace file with 62 MB size and even then that processing time is smaller than the time it takes the other two tools to process the trace file with 2 MB size.

9. CONCLUSION

In this chapter we presented a new tool named TRAFIL that firstly aims to assist and support the procedure of analyzing simulation trace files and secondly to automate the execution of NS-2 simulations along with the execution and analysis of video simulations. To accomplish these goals TRAFIL introduces the novel idea of using metafiles and sub metafiles that renders the tool and the process of identifying trace file types more abstract and robust. The task of creating a metafile and adding it to TRAFIL's metafile and sub metafile repository is trivial; therefore TRAFIL is independent of trace file format and can be used with a variety of trace file formats. Moreover, one of the main objectives of TRAFIL was to speed up the identification and processing phases of opening a trace file. Similar tools have not been very effective during this task and needed a fair amount of time to open a trace file. As we thoroughly presented in section 8 the use of metafiles and sub metafiles enabled TRAFIL to carry out this task in significantly reduced time compared to other popular tools (up to 100 times faster). Another unique characteristic of TRAFIL in regard to the trace file analysis domain was the ability to store each trace file in a local database, alleviating this way the wearing task of having to re-open each trace file from the disk. Furthermore, TRAFIL gave the opportunity to retrieve a variety of simulation statistics, information, QoS measurements and charts. Every single piece of information that was produced could also be exported from the tool including the actual trace files in their parsed form either in txt or Excel file.

Apart from the analysis of trace files, TRAFIL gave the opportunity to execute OTcl simulation scripts in order to have the results passed to TRAFIL immediately and kept organized without needing any further involvement. This feature although useful serves more as a utility functionality in the grater procedure of executing video transmission simulations. NS-2 and Evalvid-RA have been used extensively for video simulations, therefore with the development of TRAFIL we aspired to automate the procedure starting from the video pre processing until the production of QoS measurements and evaluation of the simulation.

10. FUTURE WORK

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

[1] TRAFIL website, http://ru6.cti.gr/ru6/research_tools.php#TRAFIL

[2] Juliana Freitag Borin, Nelson L.S. da Fonseca, 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.

[3] Network Simulation Cradle: http://www.wand.net.nz/~stj2/nsc/software.html

[4] NS2Measure: http://cng1.iet.unipi.it/wiki/index.php/Ns2measure

[5] J. Malek and K. Nowak, September 2003. Trace graph-data presentation system for network simulator ns. In Proceedings of the: Information Systems - Concepts, Tools and Applications (ISAT 2003), Poland.

[6] The Trace File analysis tool Trace Graph: http://www.angelfire.com/al4/esorkor/

[7] The Mathworks, Inc., 3 Apple Hill Drive, Natick, MA 01760-2098,USA.

[8] Qian, H. and Fang, W. (2008). Jtrana: A java-based ns2 wireless trace analyzer: http://sites.google.com/site/ns2trana/

[9] Aliff Umair Salleh, Zulkifli Ishak, Norashida Md. Din and Md Zaini Jamaludin, 27-28 June, 2006. Trace Analyzer for NS-2. In Proceedings of the 4th Student Conference on Research and Development (SCOReD 2006), Shah Alam, Selangor, MALAYSIA.

[10] The Trace File analysis tool Trace Analyzer: http://trace-analyzer.sourceforge.net/

[11] Ahmed Sobeih , Wei-Peng Chen , Jennifer C. Hou , Lu-Chuan Kung , Ning Li , Hyuk Lim , Hung-Ying Tyan , Honghai Zhang, J-Sim: A Simulation Environment for Wireless Sensor Networks, Proceedings of the 38th annual Symposium on Simulation, p.175-187, April 04-06, 2005

[12] J-Sim, Available on: http://sites.google.com/site/jsimofficial/

[13] Lie A, Klaue J., Evalvid-RA: Trace Driven simulation of rate adaptive MPEG-4 VBR video, Multimedia Systems 2008; 14(1): 33-50.

[14] Evalvid-RA website: http://www.item.ntnu.no/~arnelie/Evalvid-RA.htm

[15] Christos Bouras, Savvas Charalambides, Georgios Kioumourtzis, Kostas Stamos. TRAFIL: A tool for enhancing simulation TRAce FILes processing. DCNET 2012 Rome, Italy.

[16] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, 2003. “RTP: A Transport Protocol for Real-Time Applications”, RFC 3550, July 2003.

[17] Lie A., “Trace driven simulation of rate adaptive MPEG-4 video” http://www.item.ntnu.no/~arnelie/evalvid_test/Presentation_Dec05.pdf

[18] Ryad Ben-El-Kezadri , Farouk Kamoun , Guy Pujolle, October 27-31, 2008. XAV: a fast and flexible tracing framework for network simulation, Proceedings of the 11th in-

ternational symposium on Modeling, analysis and simulation of wireless and mobile systems, Vancouver, British Columbia, Canada