Simulation Modelling Practice and Theory 30 (2013) 90–110

Contents lists available at SciVerse ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

# A tool for automating network simulation and processing tracing data files

Christos Bouras [a,b,*], Savvas Charalambides [a,b], Michalis Drakoulelis [b], Georgios Kioumourtzis [d], Kostas Stamos [a,b,c]

[a] Computer Technology Institute and Press, N. Kazantzaki Str., University Campus, 26504 Rio, Greece
[b] Computer Engineering and Informatics Department, University of Patras, Greece
[c] Technological Educational, Institute of Patra, Greece
[d] Center for Security Studies, P. Kanellopoulou 4, T.K. 10177, Athens, Greece

ARTICLE INFO

*Article history:*
Received 23 May 2012
Received in revised form 24 September 2012
Accepted 25 September 2012
Available online 25 October 2012

*Keywords:*
Simulation analysis tool
Trace file
NS-2

ABSTRACT

One of the most highly regarded discrete event simulators used for network simulation is NS-2 (Network Simulator). NS-2 executes simulation scenarios producing various data of which trace files are considered the most beneficial for evaluating a simulation. In this paper we present the design considerations and implementation of a new tool that can be used both in TRAce FILe analysis and execution of simulations using NS-2. Although TRAFIL is primarily based on NS-2 trace files it can be extended to support a number of different other simulation trace file formats. It 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. In order to accomplish the above 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.

© 2012 Elsevier B.V. All rights reserved.

## 1. Introduction

Computer and telecommunication networks simulation [1,2] is a method where a tool is employed to model the behavior of an actual network. In most cases the simulation of a network involves the computation and the monitoring of the interaction between network elements as well as the analysis and process of the observations. One of the most important aspects

* Corresponding author at: Computer Engineering and Informatics Department, University of Patras, Greece. Tel.: +30 2610996951.
*E-mail addresses:* bouras@cti.gr (C. Bouras), charalampi@ceid.upatras.gr (S. Charalambides), drakouleli@ceid.upatras.gr (M. Drakoulelis), kioumourtzis.kemea@gmail.com (G. Kioumourtzis), stamos@cti.gr (K. Stamos).

1569-190X/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.simpat.2012.09.001

of network simulation is the opportunity to alter the features of a network model in order to produce new data for analysis and compare them with previous simulation efforts. Changing the model of a network is obviously not as time consuming and costly as having to alter a real network's parameters in order to evaluate its performance under various scenarios.

Network simulation serves the needs of researchers and developers to design and analyze networks efficiently and with low cost and it is made possible via Network simulation tools [3–5]. There are numerous network simulators both commercial and open source like OPNET, OMNet++ and Network Simulator 2 (NS-2) [6]. NS-2 [7,8] is one of the most widely used network simulators [9]; it offers support for various protocols like TCP, UDP and different routing and multicast protocols. NS-2 users define the network topology and simulation scenario that is executed by the main NS-2 module. During the simulation NS-2 enables the monitoring of a number of different network elements as well as monitoring of the entire network traffic which is recorded in special files called Trace Files.

Trace files [10] are considered some of the most important information a Network Simulator has to offer. They describe all the events that have taken place during a simulation and users can retrieve valuable feedback about the networks' behavior based on these events. There are three different trace file formats that NS-2 produces depending on the simulation scenario. These formats are Normal trace file, Old Wireless trace file and New Wireless trace file. The first trace file type records information for simulations of wired networks and the latter two are the trace files produced for wireless network simulations. TRAFIL can be used to process each one of these three trace file types.

The motivation for this work derives from the fact that the ease and efficiency that NS-2 offers in conducting large scale simulations and the ability it gives to test various protocols and assumptions is not found in the post simulation phase. The produced trace files present the actual information and what took place during the simulation. Thus, their process is to some extent as important as the actual simulation. Nevertheless, the analysis of trace files constitutes a phase in which a user has to strive to obtain the results he wants as every simulation has different objectives and every researcher aims to obtain different results. In addition, the fact that trace files even for small sized simulations tend to have great volume makes their management and analysis a wearying task.

The tool we propose, as we have mentioned, accepts any of the NS-2 produced trace files and stores them at a local database in a more suitable for analysis form. After having stored the information of a trace file the user has the opportunity to retrieve a great number of statistics and charts regarding a network's behavior such as End-to-End Delay, Packet Loss, Throughput, Delay Jitter, etc. Thus, by developing TRAFIL we aspire to present a tool that will simplify and speed-up the post simulation analysis of trace files.

Furthermore, TRAFIL [11] aims to automate the whole simulation procedure by enabling the user to input a script file that describes the simulation in order to execute it. Secondly, TRAFIL facilitates the execution of video transmission simulations using the Evalvid-RA module.

Finally TRAFIL supports user initiated SQL queries. Every trace file processed by the tool is stored to a local MySQL database in a parsed form so that the data can be used to produce measurements. During the development of TRAFIL we aimed to offer the opportunity to retrieve the most important information and QoS parameters. Nevertheless, a user could be interested in specific information that for the time being TRAFIL does not offer. Consequently, in order to address this possibility we provide the opportunity for a user to initiate queries directly to the local database, as well as, export each trace file.

The remainder of this paper is organized as follows: In Section 2 we present similar work that has been done and briefly explain how TRAFIL compares and what we aim to offer that is not currently present. In Section 3 we introduce TRAFIL, present its architecture and explain the design issues that arose during the development and how we solved them. Section 4 analyzes key points in TRAFIL's implementation and presents the core procedure of identifying a trace file's format. In Section 5 we describe all the different functionalities that TRAFIL offers to facilitate the analysis of a trace file and in Section 6 we illustrate their actual utilization using a sample simulation trace file. We conclude the paper in Section 7 and describe our plans for future work in Section 8. TRAFIL can be found publicly available at [31].

## 2. Related work

Similar work [12] and tools have been developed that produce statistics of a simulated network's behavior. Some of these projects [13,14] 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 [15,16] 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 [17] 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 [18] is a Java based NS-2 wireless trace analyzer. It can be used to analyze 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 [19,20] 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 [21,22] 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 developed initially to aid the analysis of network simulation trace files and its architecture is depicted in Fig. 1. In this section an analysis of the design issues that arose during the development of TRAFIL will be presented along

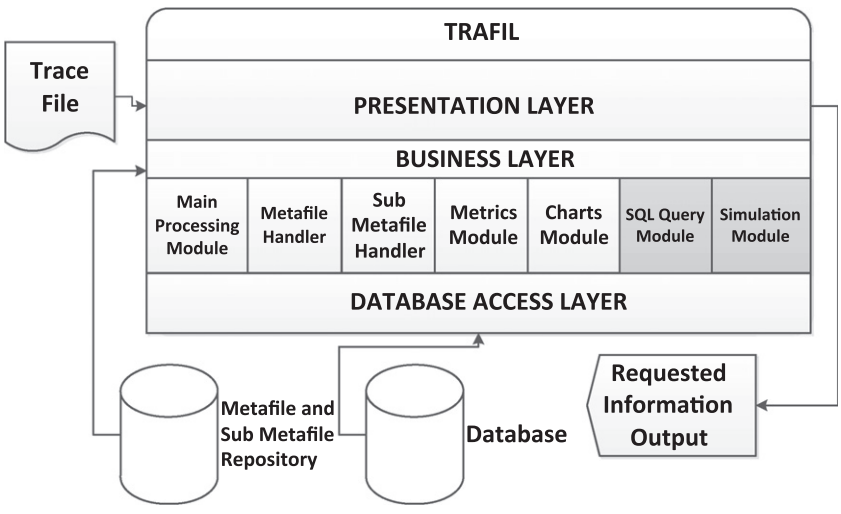


**Fig. 1.** TRAFIL architecture.

with how they were alleviated. In addition, the tool's architecture will be analyzed as it was formed based on the objectives it was designed for as well as the issues that occurred during the development phase.

### 3.1. Design issues

One of the main goals of TRAFIL is to aid the analysis of trace files and for now the tool currently focuses on NS-2 trace files. As we have mentioned earlier there are three main trace file formats that are produced by NS-2 which are:

- Normal trace file.
- Old Wireless trace file.
- New Wireless trace file.

Normal and Old Wireless trace files have a static format. Namely, they both start each line with a special character that describes the event that has taken place which is followed by information regarding that event. Although a New Wireless trace file also begins each line with an event indicator, it contains an arbitrary number of fields in each line to describe an event.

Consequently, a very straightforward way of implementing the trace file recognition process could be hard coding each possible trace file structure inside the tool's source code. This way, when a trace file is given as an input we can attempt to match its structure with one of the hard coded formats. However, this is not a very generic way of implementing the trace file recognition phase and it becomes obvious that it comes with a serious drawback. If at some point in time NS-2 changes a trace file's format then the tool will become useless because it won't be able to recognize that trace file. In order for it to work it will need serious rebuilt of the code. To tackle the issue of trace file recognition during the trace file analysis procedure TRAFIL utilizes the notion of metafiles, which to our better knowledge has not been employed by any other tool used for trace file analysis. Metafiles can be defined as specific utility files; they contain data used to describe other data. In that point of view, metafiles are mainly employed by TRAFIL to describe the structure of each one of the three aforementioned trace file formats. As will be described later, metafiles are also used to aid the trace file analysis and measurement production procedures apart from identifying the format of a trace file.

Furthermore, each of the three trace file formats mentioned earlier contains a number of extra header fields depending on the simulation scenario and the kinds of packets that travel through the simulated network. Each trace file format has a specific number of header fields that can be added in each of the events it logs. In fact each line of a trace file can contain a different number of header fields and it might also contain some of the same header fields but in different order. It becomes apparent that the number of combinations of header fields that might be present inside a trace file is a large one. In order to cope with this issue TRAFIL introduces the concept of sub metafiles. The basic idea is that since each trace file contains a number of different header fields to describe a specific kind of packet it is only logical for TRAFIL to do the same. Therefore, for each different header type that a trace file of a specific format uses, TRAFIL has a corresponding sub metafile to describe its structure. As will be shown in the next sections the structure of the sub metafiles follows the same conventions of that of metafiles and their use enables TRAFIL to compensate with any different trace file that might be given as an input.

Finally, another issue that appeared during the design and implementation of TRAFIL was the transfer of data to the local database for storage. It is not rare for a simulation to produce a trace file with great size and as a result the manner in which its transfer to the database is implemented becomes an issue. The trace file identification phase of TRAFIL with the use of metafiles and sub metafiles is extremely efficient. Thus to render the complete analysis phase equally efficient we used methods of data transfer to the local database that yield the lowest possible transfer time.

### 3.2. TRAFIL architecture

TRAFIL is built on a three-layer architecture, as can be seen in Fig. 1. The first layer is the Presentation Layer which is responsible for handling the user interaction. This layer implements the graphical interface from which the user is able to interact with TRAFIL; the user's requests are passed by the Presentation Layer to the Business Layer where the actual processing and calculation takes place.

The Business Layer handles the user's requests and conducts all the actual process. In order to facilitate all the different operations that TRAFIL conducts, the Business Layer consists of a number of different modules each of which is responsible for a different task. The Main Processing Module is responsible for the trace file identification, parsing and storage task which is one of TRAFIL's core procedures. It is also responsible for calculating the General Simulation Information and the General Node Information. The Metafile Handler and Sub Metafile Handler modules are responsible for loading and validating the metafiles and sub metafiles respectively. Measurement calculation is performed by the Metrics module and the Charts module is responsible for plotting each different chart type. The SQL Query Module handles the queries that are issued by a user to the local database and finally the Simulation Module conducts all the appropriate actions in order to enable users to execute Network or Video transmission Simulations.

The Database access layer is responsible for storing data to the local database or returning the data requested by the Business Layer. The Database access layer interacts only with the Business layer form which it either receives data destined for storage or returns data from the local database.

*3.2.1. Database*

The Database component of TRAFIL is a MySQL database where all information about each trace file is stored. The contents of the database include one table for every trace file the user has inputted and each table is named after the trace file whose data it contains. As we have stated earlier there are three different trace file formats produced by NS-2 and also other simulators might produce trace files that have completely different structures than NS-2 trace files. It is evident that not all tables can have the same structure and that a table's structure depends heavily on the trace file's format.

Thus, we use the metafiles and their sub metafiles to create a table for a trace file. A metafile contains the structure of a trace file as will be explained in more detail in the following sections. Using a metafile that was matched to a trace file we can create a table inside which we will be able to transfer the trace file's data. The table's structure depends also by the metafile's sub metafiles since they too can be used at some point during the trace file analysis.

In addition to the tables who contain trace file data, there is also a table that contains the standard measurements that are produced when a trace file is loaded into TRAFIL. For every trace file loaded into TRAFIL there is a record in that table, this way the tool does not have to create the measurements each time a user loads a trace file from the database. On the contrary, as with the trace file's information, the standard measurements are also loaded from the database minimizing this way the loading time.

*3.2.2. Metafile and sub-metafile repository*

The metafile and sub metafile repository is the part of TRAFIL that contains all the 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.

The repository contains all the metafiles for the three known NS-2 trace file formats with all their required sub metafiles. This way TRAFIL can be utilized with every NS-2 trace file. If a user wants to process a different kind of trace file he simply can create a metafile that describes this new trace file format and place it in the metafile repository along with all the other metafiles. Furthermore, if this new trace file type supports different kinds of header fields the creation of appropriate sub metafiles would suffice to render TRAFIL able to process the information.

## 4. Implementation issues

In this section we describe the most notable and unique features of TRAFIL. We introduce and discuss the structure and use of metafiles and sub metafiles. In addition, we explain one of TRAFIL's core operations, the trace file identification, as it is implemented in a novel manner utilizing the concepts of metafiles and sub metafiles.

### *4.1. 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 Fig. 2. As Fig. 2 shows there is 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.

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 NumberOfFields is used to ensure that the metafile's structure is correct. The value defined by the NumberOfFields 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.

```
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
```

**Fig. 2.** Metafile structure.

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 Fig. 2.

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 signaled 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 traveling 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:

Fig. 3 depicts the structure of the sub metafile used to represent a satellite packet's header information which is logged in a Normal trace file. Fig. 4 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.

### 4.3. Trace file recognition process

The trace file recognition process is one of the most important procedures of TRAFIL and the way it is implemented distinguishes our tool from other similar tools. In this particular phase the use of metafiles and sub metafiles comes to place and we explain how they are used.

TRAFIL enables users to extract information from trace files as well as calculate measurements and plot charts. In order though for it to be able to analyze the data contained inside a trace file it must identify its format first. The general idea of the mechanism for the analysis, identification and storage of a trace file is shown in Fig. 5 and it is explained in the remainder of this sub section.

The trace file identification process is the procedure in which TRAFIL attempts to identify in which of the three categories the trace file that was given as an input belongs. Based on the encoding that is described in all the metafiles of TRAFIL's metafile repository the tool tries to find a match between a metafile and the trace file.

When starting the trace file identification process all the metafiles are loaded and for each new line that is read from the trace file we try to match its structure with that which is described by a metafile (e.g. Normal trace file, Old Wireless). We try to match all the unique characters that are present in the metafile with the corresponding fields in the line read. The process continues for every line of the trace file until a match is established. If no match is established possibly the structure of the trace file is not in the correct format or no metafile has been constructed to describe it.

```
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 .
```

**Fig. 3.** 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
```

**Fig. 4.** TCP sub metafile.

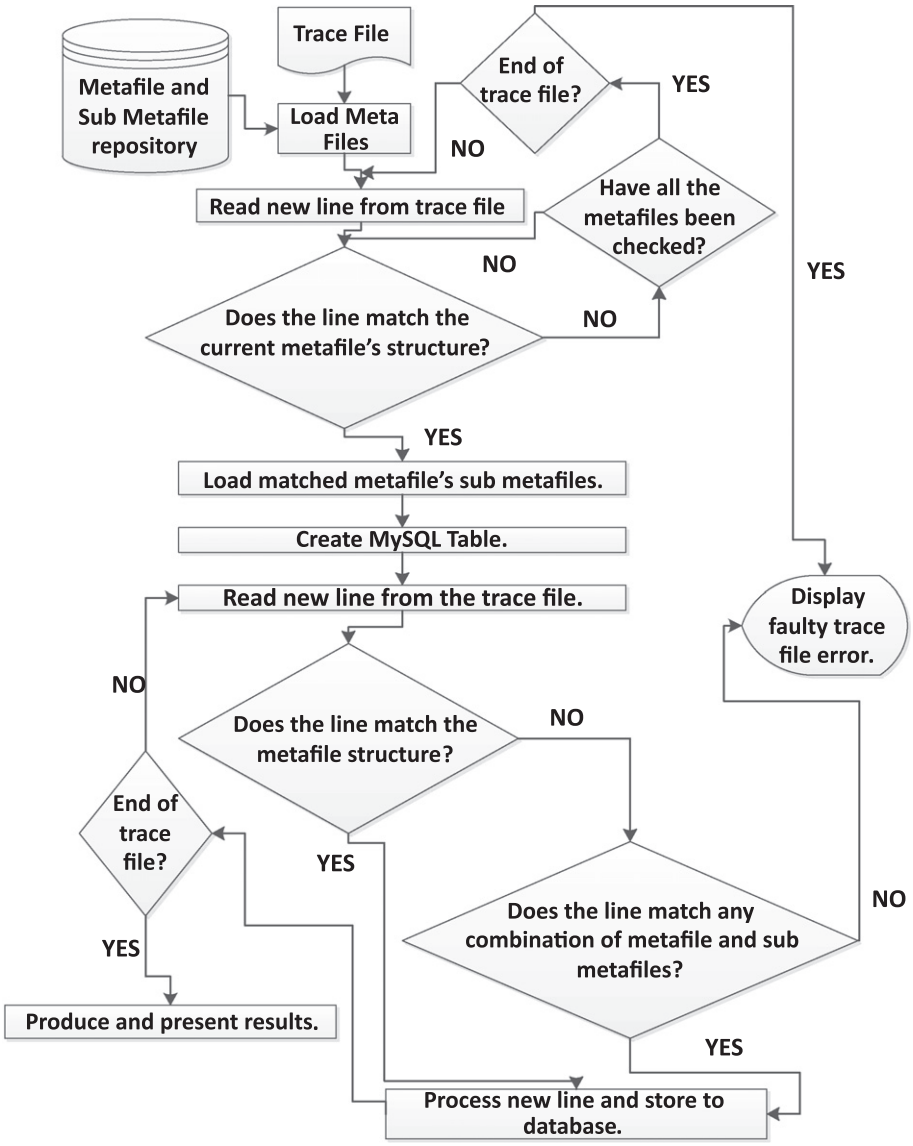


**Fig. 5.** Trace file identification and data transfer procedure.

If the trace file has eventually been matched to a metafile, all sub metafiles of that metafile are loaded and the data transfer process can begin. Prior to the transfer, a table is created in the local database. The name of the table is the name of the trace file that was given as an input. This table will have as many columns as all the different fields that are defined in the metafile and its utility sub metafiles, namely all the fields that describe columns (start with the -name flag). After the table is created the data transfer process can begin.

In the data transfer process the trace file is read line by line. If a line matches the metafile or a number of sub metafiles as well, depends firstly on the number of elements that the line has. If the number of elements in a line is the same as the metafile's NumberOfColumns, then it is for sure a line that is described only by the metafile. We then use the metafile to process the line, extract the information and store it to the database.

The only other possibility is that the line has more fields than the metafile structure. If this is not the case then the structure of the trace file is wrong and the data transfer procedure is stopped.

If the number of elements in a line is greater than that which is defined by the metafile's NumberOfColumns then the sub metafiles are used. We start by attempting to match a sub-metafile with the remaining elements that exceed the number of elements that are described in the metafile's structure. For each sub metafile that satisfies the condition of having less or equal elements than the remaining number of elements we try to match all of the unique flags it contains. If we find a sub-metafile that matches a specific set of elements both in the number and the unique flags we bind that range of elements

```
WHILE  newline IS NOT EMPTY
        IF  newline's length EQUALS metaFile's  NumberOfColumns  value
            IF newline's Unique Characters EQUALS metaFile's Unique Characters
                        lineData = process(newline)
                        store(lineData)
            END IF
        ELSE IF newline's length > metaFile's  NumberOfColumns  value
                        columnLength = metaFile's  NumberOfColumns  value
                        lineData = process(newline)
                        residueData = newline - lineData
                        WHILE  residueData's length  > 0
                                FOR each subMetaFile in the sub metafile repository
                                        IF  residueData's length >= subMetaFile's NumberOfColumns  value
                                                IF residueData Unique characters EQUALS the subMetaFile's Unique characters
                                                        lineData = lineData + process(residueData)
                                                        update(residueData)
                                                        update(SubMetaFileList)
                                                END IF
                                        END IF
                                END FOR
                                IF residueData's length has not changed:
                                        show( error( “New Line does not match any combination.” ) )
                                END IF
                        END WHILE
                        store(lineData)
        ELSE
            show( error ( “New Line does not match any metafile.” ) )

        END IF

END WHILE
```

**Fig. 6.** Trace file data transfer mechanism.

to that sub metafile. We repeat this task for all the remaining elements of that line and the remaining sub-metafiles until we match all the elements to sub-metafiles.

If we reach a point that a range of elements in a line cannot be matched to any sub metafile there must be either an error in the trace file structure or no available sub-metafile exists that can describe this group of elements. In either case the data transfer procedure cannot continue and thus stops.

The data transfer mechanism that takes place after the trace file has been matched to a metafile as described above requires the use of both the metafiles and the sub metafiles. It is one of the most important procedures of TRAFIL and therefore we give a more detailed explanation of it in Fig. 6.

## 5. Trace file analysis

This section gives a brief description of the core procedures and functionalities offered by TRAFIL. A summary of previously analyzed issues is made in some cases i.e. identification of a trace file and in some other cases new functionalities that only TRAFIL offers are presented like the opportunity to conduct a complete video transfer simulation.

### *5.1. Identification and storage*

The first stage of the trace file analysis procedure is the identification of the trace file that has been given as an input. When a trace file has been selected, TRAFIL firstly tries to identify the metafile that can be used to process its contents. In other words the tool finds the metafile whose encoding matches the structure of the input. If no metafile describes the input then the trace file structure is either wrong or no metafile has been created to support that kind of input.

When a match has been established, namely the trace file has been identified; we proceed to the parsing and storage phase. The first step that is taken to start the parsing of the input is to load all the corresponding sub metafiles of the primary metafile which was matched with the input. Following this task the trace file is read line by line, parsed based on the primary metafile and its sub metafiles and when we eventually reach the end of the input we start the data transfer procedure. If at any point of the trace file parsing process a great number of errors is found in the trace file structure the procedure stops. TRAFIL tries to ignore errors, but if the number of lines that have been identified as wrong exceed fifty the data that were parsed up to that point are discarded and the user is notified of the errors that occurred.

Finally, when the trace file processing and parsing phase is over the information extracted must be stored to the local database. Each trace file that is given as input has its own table named after it and the information regarding it is stored there.

### 5.2. Statistics calculation

The trace file processing and storage procedures are followed by the phase in which TRAFIL calculates all the general simulation information regarding the input trace file. These metrics are calculated automatically without any user involvement. Other information that refer to specific nodes and measurements that describe QoS parameters of the simulation are calculated only after a user request. An example of both occasions is featured in the next section.

#### 5.2.1. General simulation information

The general simulation information that is calculated regarding the input involves information such as start time, end time and overall time of the simulation. When it comes to the nodes of the simulation the information that is presented are the Number of Nodes and Number of Sending Nodes. As for the information relating to the packets and bytes that were transferred or dropped during the simulation the information that is logged is the 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.

Finally, for 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. This way a user that might be interested only in the packets and bytes for a specific layer of communication can get the information he wants.

#### 5.2.2. Node specific information

Besides the general simulation information that is calculated by TRAFIL the user can additionally extract explicit information regarding a specific node of the simulation. After the trace file is processed and the simulation information is generated the user can produce similar information by selecting a specific node. The information that can be produced includes statistics such as Number of Sent, Received, Dropped, Forwarded and Generated Packets and Bytes for that node.

Moreover, for wireless scenario simulations TRAFIL provides additional metrics for the AGT, RTR and MAC trace levels as it does for the general simulation information.

#### 5.2.3. QoS parameters

The QoS parameters that can be calculated via TRAFIL for a specific trace file are Packet Delivery Rate in packets/s as well as Throughput in bytes/s, Minimum End to End Delay, Maximum End to End Delay, Average End to End Delay, Delay Jitter, Minimum Delay Jitter, Maximum Delay Jitter, Average Delay Jitter and Packet Loss Ratio. In order to retrieve these measurements the start node and end node must be specified for which the communication will be calculated. The Delay Jitter related measurements are calculated based on the RFC 3550 [23] 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 three 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.

### 5.3. Plotting charts

Following the trace file processing phase the user can plot numerous charts based on the information of the inputted trace file. The plotting of a chart is enabled through the Charts tab of TRAFIL and there the user can set the parameters to create his charts. The charts offered by TRAFIL are Packet Delivery Rate in packets/s as well as Throughput in bytes/s, Delay Jitter and Packet Loss Ratio. These charts can be drawn either between two specified nodes or by choosing only one node. Other parameters that the user must select in order to draw a chart are the sampling rate in seconds and the trace level in which the communication between the two nodes must be retrieved. After all the parameters have been set then the chart can be plotted, if there is no communication in the communication level specified then an empty chart will be displayed.

#### 5.3.1. Node to node charts

If the user wants to plot a chart regarding the communication between two nodes then he can do so by selecting the start node and end node he prefers. After setting these two parameters, then he must set an appropriate trace level. For wired simulations the appropriate layers are Link Layer and Physical Layer and for wireless scenarios the appropriate trace levels are AGT, MAC and AGT. Finally he can select a sampling rate that is either one or five seconds.

#### 5.3.2. Node specific charts

The other option for a chart is to select only a single node. The same parameters as with the previous chart type must be set, namely the level of communication and the sampling rate. After the parameters, a specific node and the chart type have been set, the plotted chart is presented.

### 5.4. User initiated SQL queries

TRAFIL has its own local database where every trace file's information is stored. The structure of each trace file's table is based on the metafile it was matched during the identification phase. The columns of the table are as many as the different fields of the metafile and all its corresponding sub metafiles. These tables are used both for storing and calculating purposes. Having all the information of each trace file permanently stored in a local database enables their reuse on demand without having to reload them from the discolor.

Although the general simulation information and metrics that are produced by TRAFIL cover the most common statistics that a user might want about a simulation there is always the possibility that someone is interested in a specific parameter or information that for the time being is not offered by TRAFIL. That is why the tool enables the user to initiate SQL queries directly to the database.

### 5.5. Simulation execution

TRAFIL is not only a tool that can be used to store and process trace files; it also enables a user to execute OTcl scripts. Until now a user in order to execute a simulation he had to run NS-2 and then manually extract information usually with the use of bash scripting or other spreadsheet software. TRAFIL gives the user the ability to locate the OTcl script he wants to execute, give it as an input and then TRAFIL executes this scenario by invoking NS-2.

The resulting trace file is then analyzed by invoking the whole procedure that was described in Section 4. This way the trace file is produced, parsed, analyzed and stored in the local database without requiring any further user involvement.

### 5.6. Video transmission simulation using Evalvid-RA

One of the most important uses of NS-2 is for the simulation of video transmission and some frameworks have been created to support this functionality. Evalvid-RA [24,25] is among the most popular frameworks, it is based on Evalvid [26] and enables a user to process a video in order to transform it to a more suitable or desired form for transmission.

After the video has been properly processed (using the FFmpeg and MP4 tools), NS-2 is used to simulate its transmission via a user defined network topology. The results of the simulation can be used to draw conclusions and evaluate any mechanism that might have been employed to enhance the video transfer. 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. Usually a trace file includes data for the frame type and size, fragmentation into segments, frame number and timing for each video frame. One of the advantages of Evalvid-RA is that it avoids the time consuming task of using the actual binary multimedia content to conduct the multimedia transfer; on the contrary it uses the generated trace files which have already been created and thus speeds up the whole process. Finally, Evalvid-RA can rebuild the transmitted video as it would have been received

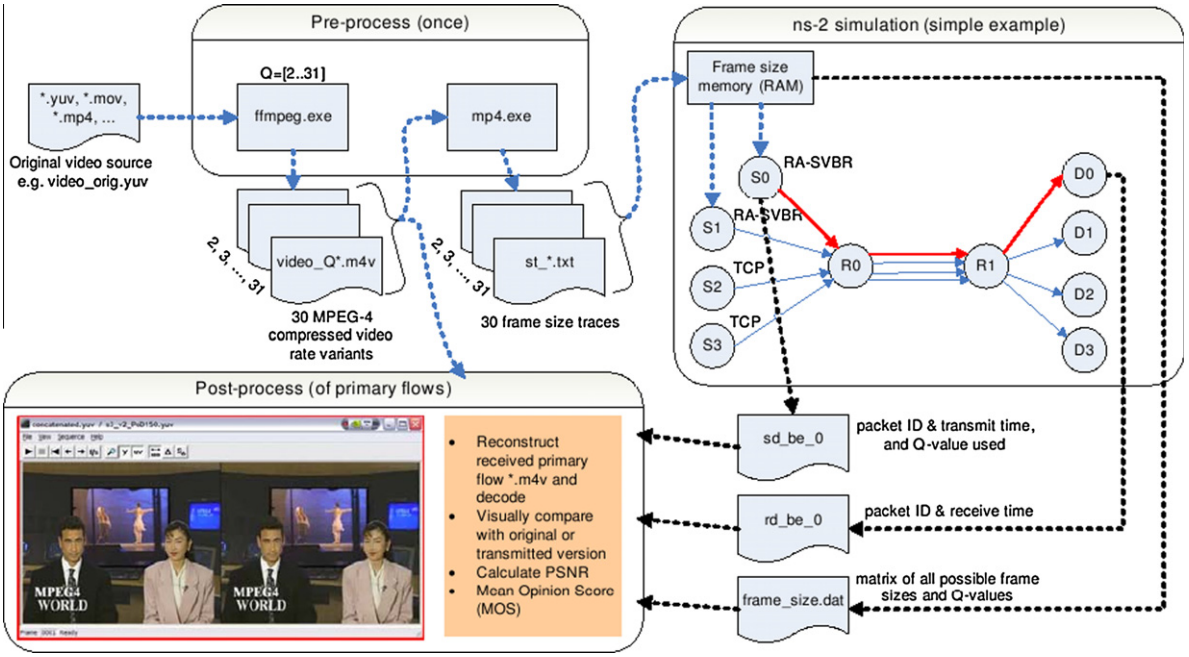


**Fig. 7.** Evalvid-RA framework [25].

by an actual receiver by the combination of information produced by the simulator and the original video file. Using information produced throughout this procedure the Evalvid-RA toolset enables users to calculate the total noise introduced (PSNR in dB). A usage of Evalvid is illustrated in Haukass [27].

Evalvid-RA uses a number of smaller scale tools to perform both the pre-simulation and post-simulation phases.

During the pre-simulation phase the main tools that are incorporated are FFmpeg and MP4. FFmpeg [28,29] is an open source cross platform tool that can be used to compress or decompress video or audio in a number of different formats, it's most common application with Evalvid-RA is to convert raw .yuv video to .mp4 format with different quantization parameters. MP4 is also an open source tool that comes with the Evalvid-RA distribution and it is used to create the trace files that contain information about a specific video frame. MP4's input comes from FFmpeg and more precisely the different .mp4 videos it produces, based on these videos MP4 creates trace files which are later passed onto NS-2 to simulate the video transmission. The simulation can be executed only if Evalvid-RA has been successfully integrated to NS-2 and the results are used in the post-simulation phase. The post-simulation phase involves the et ra, fixyuv ra and psnr tools that are also included in the Evalvid-RA distribution. These tools are used to reconstruct the transmitted video from the video trace files and produce various QoS measurements. An overview of the framework and the whole process as described in this paragraph can be seen in Fig. 7.

The process of transmitting a video using this framework via NS-2 is very popular. Therefore, TRAFIL offers the ability to simulate video transmission using this framework given the fact that the interested user has successfully incorporated Evalvid-RA in his NS-2 distribution. TRAFIL uses the aforementioned tools that come with the framework's distribution to perform the pre-simulation, simulation and post-simulation phases. The user is able to define all the parameters as he would have done when executing all the steps shown in Fig. 7, and specify the simulation script he wants to test. TRAFIL then can execute the whole procedure and returns all the results in a user friendly, robust and compact manner. This way TRAFIL aims to facilitate the need for a more solid and automated approach to conducting simulations regarding video content transmission.

## 6. Trafil usages 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 four wireless nodes of which three 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 s. During this simulation node 3 moved every 50 s either closer or away from nodes 0 and 1 and finally at 300 s 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 4 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 Fig. 8. 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 columns of the table 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 nontrivial 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.

### 6.1. 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 Fig. 9.

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.

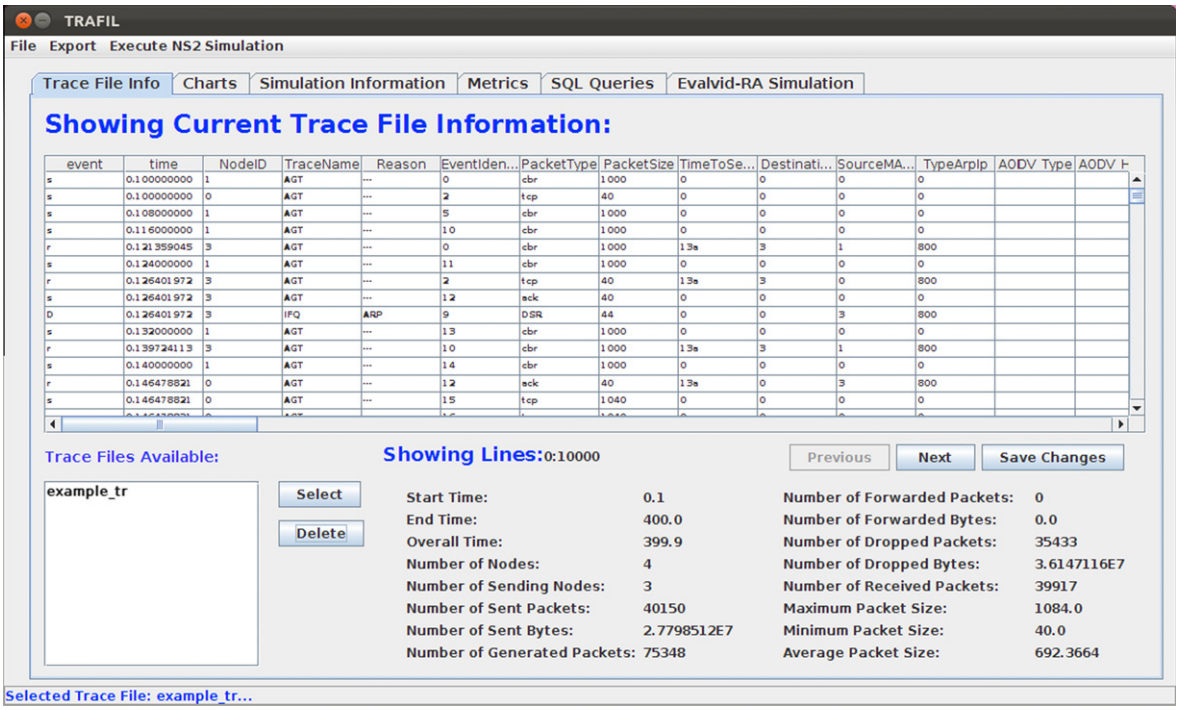


**Fig. 8.** TRAFIL main view.

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.

### *6.2. 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 Figs. 10 and 11. The charts can refer either to a pair of nodes (Fig. 10) or to a specific node (Fig. 11). 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 s. 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.

### *6.3. 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 Fig. 12.

### *6.4. NS-2 via TRAFIL*

#### *6.4.1. 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

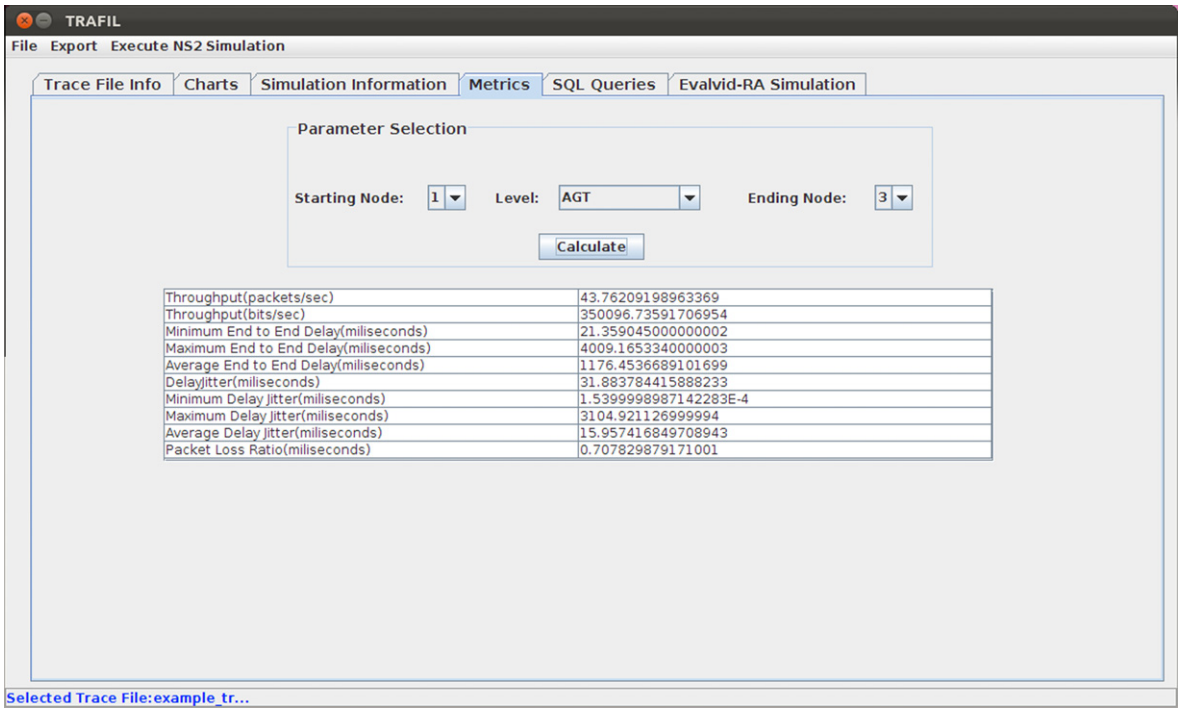


**Fig. 9.** QoS parameter extraction.

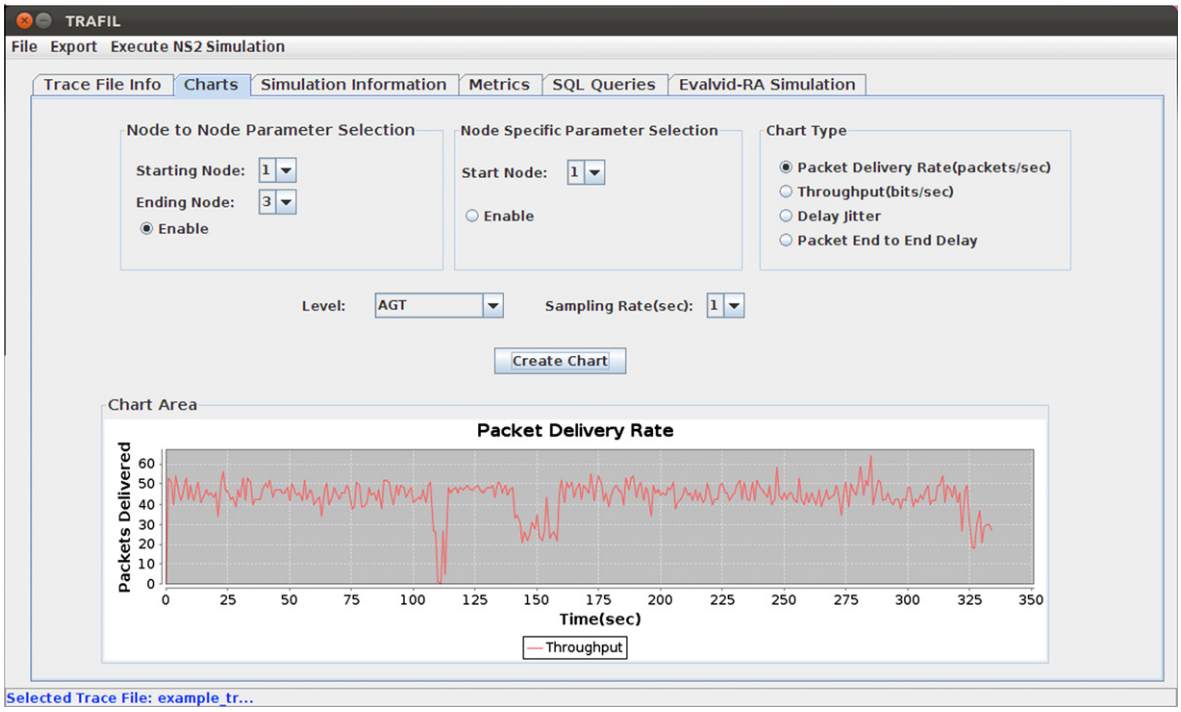


**Fig. 10.** Chart plotting.

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.

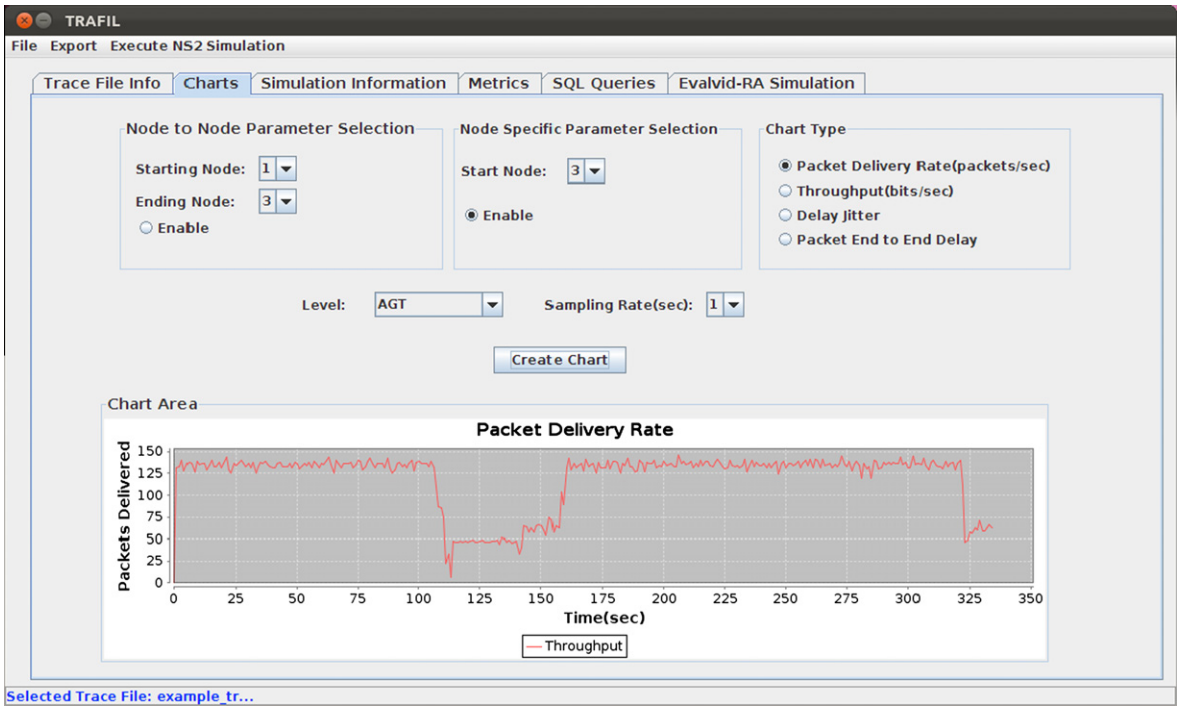


**Fig. 11.** Specific node chart.

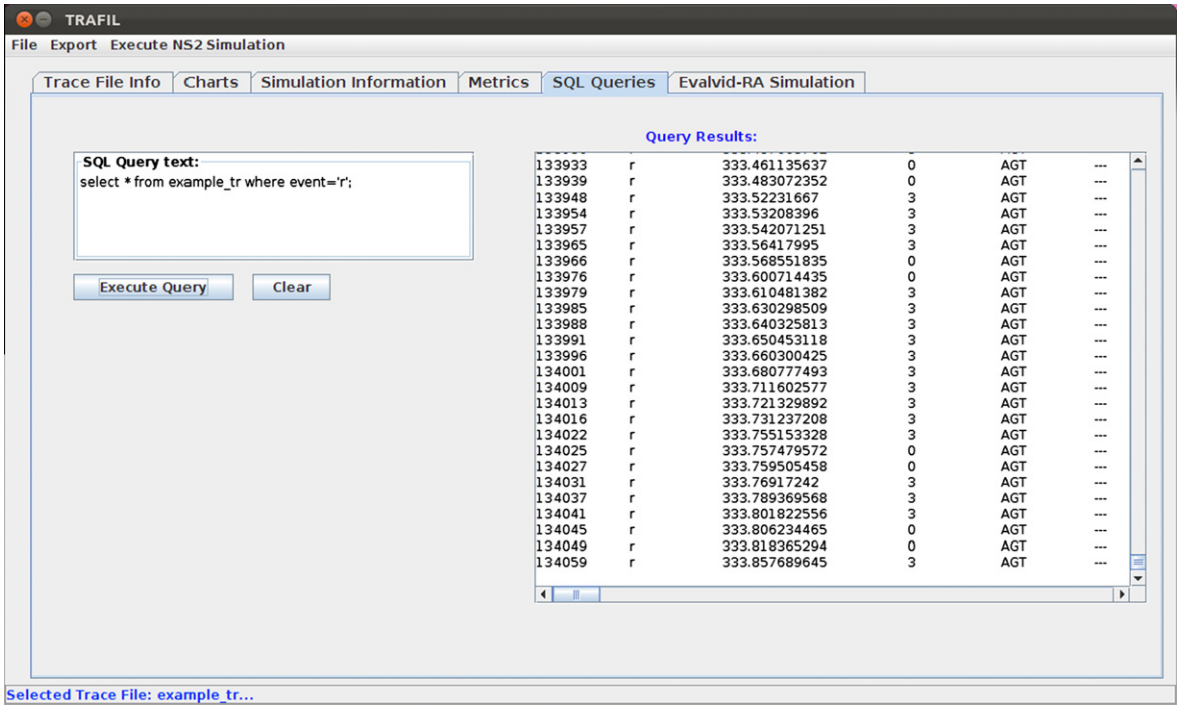


**Fig. 12.** SQL query execution.

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 4.

*6.4.2. 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 Fig. 13. 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.

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 Fig. 13. 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 Fig. 14. 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 Fig. 15. 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.

Fig. 16 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.

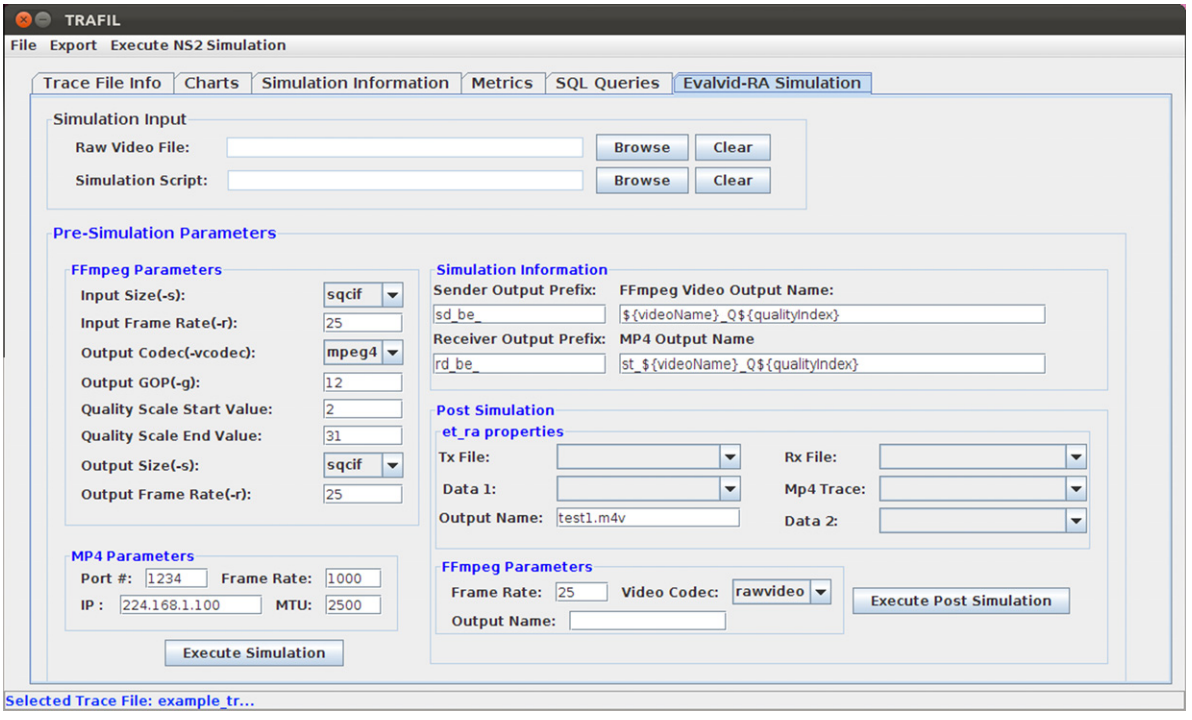


**Fig. 13.** Video simulation.

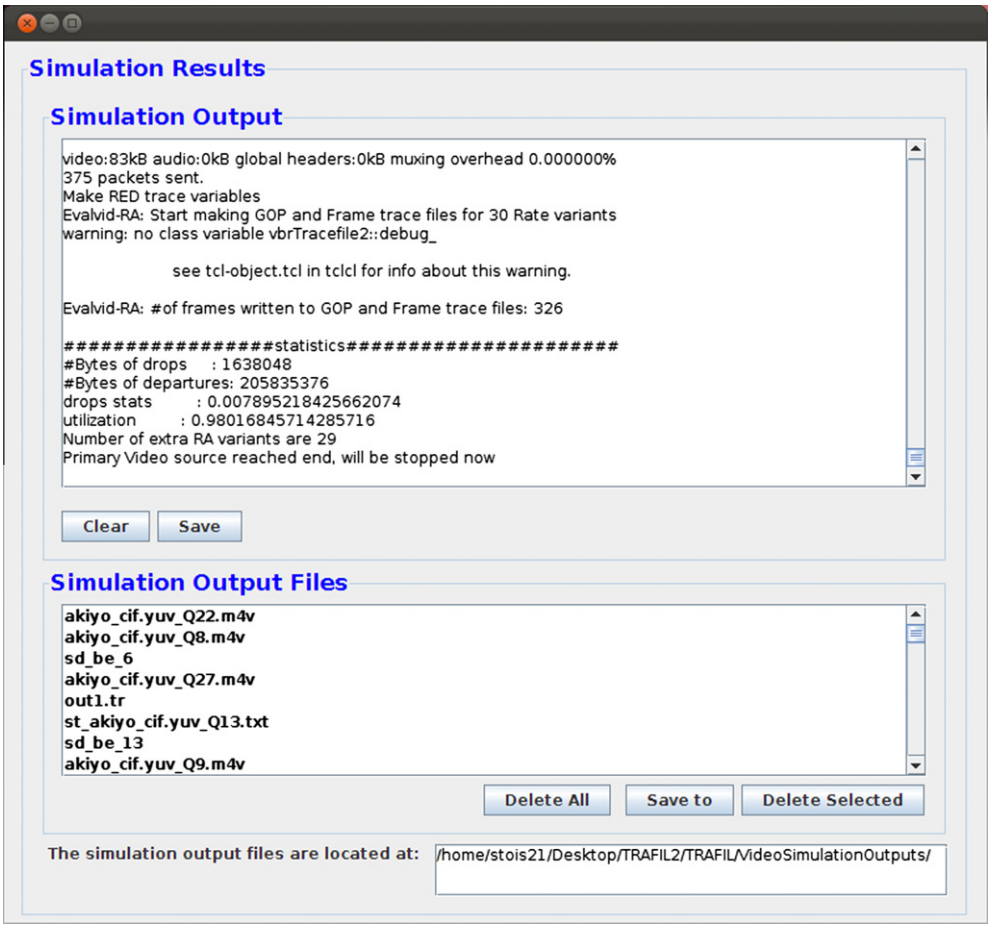


**Fig. 14.** Pre-simulation phase results.

### 6.5. TRAFIL results

In this section we present timing results about the core procedures described in Section 4 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.

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 16 MB has a smaller processing time than the 11 MB one and the reason is the trace file format. The trace file with size 11 MB was produced using a wireless scenario and the 16 MB 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 the 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.

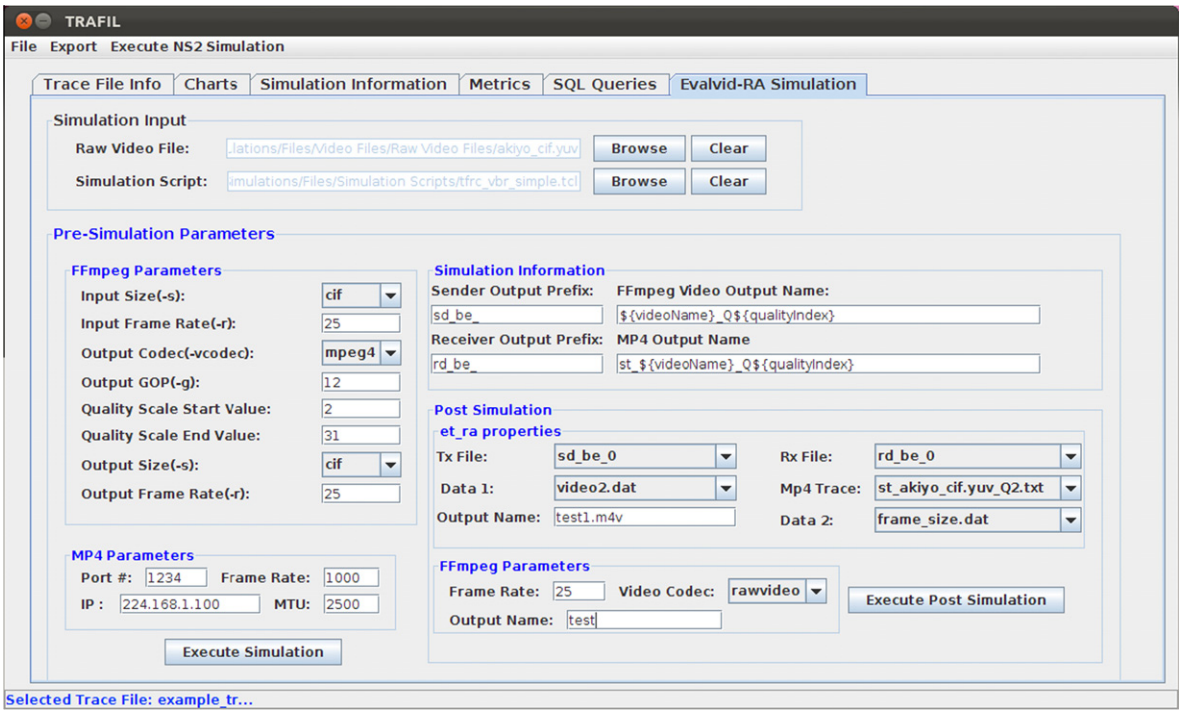


**Fig. 15.** Post-simulation phase.

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 [30]. 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:

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 accurate 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.

## 7. Conclusion

In this paper 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 6 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

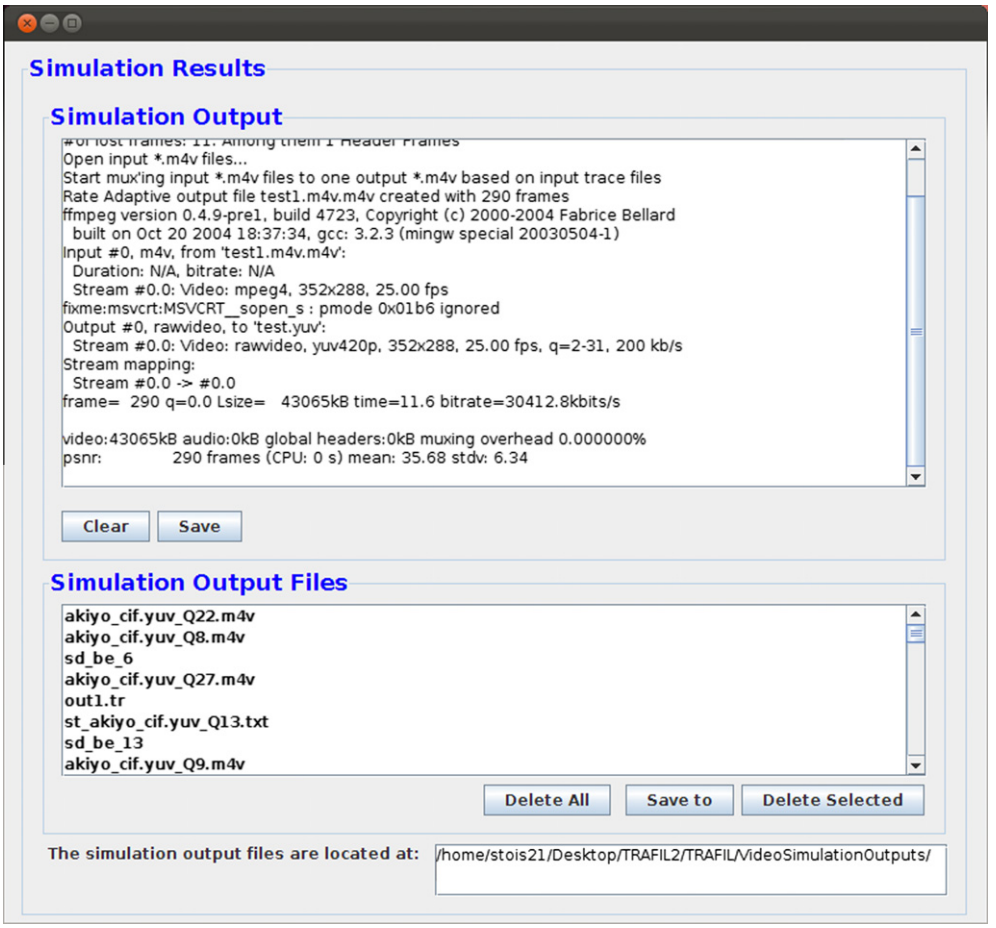


**Fig. 16.** Post-simulation results.

**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 | 1569 |
| 7 | 3558 |
| 11 | 5186 |
| 16 | 4887 |
| 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 | 1967 | 1456 |
| 6 | 2499 | 2215 |
| 13 | 4973 | 3635 |
| 24 | 8699 | 5613 |
| 35 | 13,095 | 8406 |
| 47 | 17,816 | 11,767 |
| 62 | 20,906 | 16,048 |

**Table 3**
Trace file analysis tools comparison.

| Trace file size (MB) | TRAFIL processing time (ms) | Tracegraph processing time (ms) | jTrana processing time (ms) |
|---|---|---|---|
| 2 | 1967 | 27,000 | 29,700 |
| 6 | 2499 | 75,400 | 68,000 |
| 13 | 4973 | 197,600 | 135,200 |
| 24 | 8699 | 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 |

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.

## 8. Future work

In our future work we plan to extend TRAFIL by adding the functionality to design the simulation plane. TRAFIL for now succeeds in offering an easy to use framework to execute simulations and analyze the results but the simulation plane must be described using OTcl scripts. Nevertheless, users that are not familiar with OTcl and NS-2 might want a simple interface in which they can prepare simple topologies to test different scenarios without having to learn all the different parameters required by NS-2. In that sense, we will work towards creating a user friendly interface from which a simulation plane can be designed and based on this description an OTcl script can be prepared and executed using the methods that TRAFIL already provides.

As TRAFIL is written in a modular way following the object oriented programming (OOP) principles and with the use of metafiles, we believe it is fairly easy to extend it in order to support the new NS-3 simulator traces.

## References


[1] L. Breslau et al, Advances in network simulation, IEEE Computer 33 (5) (2000) 59–67.
[2] R. Fujimoto, K. Perumalla, A. Park, H. Wu, M. Am-mar, G. Riley, Large-scale network simulation: how big? how fast? in: 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003 (MASCOTS 2003), 2003, pp. 116–123.
[3] E. Weingärtner, H. vom Lehn, K. Wehrle, A performance comparison of recent network simulators, in: Proceedings of the IEEE International Conference on Communications 2009 (ICC 2009), IEEE, Dresden, Germany, 2009.
[4] Muhammad Azizur Rahman, Algirdas Pakštas, Frank Zhigang Wang, Network modelling and simulation tools, Simulation Modelling Practice and Theory 17 (6) (2009) 1011–1031. ISSN: 1569-190X, 10.1016/j.simpat.2009.02.005.
[5] S.Y. Wang, C.L. Chou, C.C. Lin, The design and implementation of the NCTUns network simulation engine, Simulation Modelling Practice and Theory 15 (1) (2007) 57–81. ISSN: 1569-190X, 10.1016/j.simpat.2006.09.013.
[6] Network Simulator NS-2. <http://www.isi.edu/nsnam/ns/>.
[7] Claudio Cicconetti, Enzo Mingozzi, Giovanni Stea, An integrated framework for enabling effective data collection and statistical analysis with ns-2, in: WNS2 '06 Proceeding from the 2006 Workshop on ns-2: The IP Network Simulator, 2006.
[8] Qi Chen, Felix Schmidt-Eisenlohr, Daniel Jiang, Marc Torrent-Moreno, Luca Delgrossi, Hannes Hartenstein, Overhaul of ieee 802.11 modeling and simulation in ns-2, in: Proceedings of the 10th ACM Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems, October 22–26, 2007, Chania, Crete Island, Greece, 2007.
[9] Juan Luis Font, Pablo Iñigo, Manuel Domínguez, José Luis Sevillano, Claudio Amaya, Analysis of source code metrics from ns-2 and ns-3 network simulators, Simulation Modelling Practice and Theory, 1569-190X 19 (5) (2011) 1330–1346, http://dx.doi.org/10.1016/j.simpat.2011.01.009.
[10] NS-2 Trace Formats. <http://nsnam.isi.edu/nsnam/index.php/NS-2_Trace_Formats>.
[11] Christos Bouras, Savvas Charalambides, Georgios Kioumourtzis, Kostas Stamos, TRAFIL: a tool for enhancing simulation TRAce FILes processing, in: Proceedings of DCNET 2012 Rome, Italy, 2012, pp.61–64.
[12] Juliana Freitag Borin, Nelson L.S. da Fonseca, Simulator for WiMAX networks, Simulation Modelling Practice and Theory, 1569-190X 16 (7) (2008) 817–833, http://dx.doi.org/10.1016/j.simpat.2008.05.002.
[13] Network Simulation Cradle. <http://www.wand.net.nz/~stj2/nsc/software.html>.
[14] NS2Measure. <http://cng1.iet.unipi.it/wiki/index.php/Ns2measure>.
[15] J. Malek, K. Nowak, Trace graph-data presentation system for network simulator ns, in: Proceedings of the Information Systems – Concepts, Tools and Applications (ISAT 2003), Poland, September 2003.
[16] The Trace File analysis tool Trace Graph. <http://www.angelfire.com/al4/esorkor/>.
[17] The Mathworks, Inc., 3 Apple Hill Drive, Natick, MA 01760-2098, USA, 2012.
[18] H. Qian, W. Fang, Jtrana: a java-based ns2 wireless trace analyzer, 2008. <http://sites.google.com/site/ns2trana/>.
[19] Aliff Umair Salleh, Zulkifli Ishak, Norashida Md. Din, Md Zaini Jamaludin, Trace Analyzer for NS-2, in: Proceedings of the 4th Student Conference on Research and Development (SCOReD 2006), Shah Alam, Selangor, Malaysia, 27–28 June, 2006.
[20] The Trace File analysis tool Trace Analyzer. <http://trace-analyzer.sourceforge.net/>.
[21] 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, in: Proceedings of the 38th annual Symposium on Simulation, April 04–06, 2005, p.175–187.
[22] J-Sim, Available on. <http://sites.google.com/site/jsimofficial/>.
[23] H. Schulzrinne et al., RTP: A Transport Protocol for Real-Time Applications, RFC 3550, July 2003.
[24] A. Lie, J. Klaue, Evalvid-RA: trace driven simulation of rate adaptive MPEG-4 VBR video, Multimedia Systems 14 (1) (2008) 33–50.
[25] EvalVid-RA. Available. <http://www.item.ntnu.no/~arnelie/Evalvid-RA.htm>.

[26] Jirka Klaue, Berthold Rathke, AdamWolisz, EvalVid – a framework for video transmission and quality evaluation, in: Proc of 13th International Teletraffic Congress (ITC'13), 2003.
[27] T. Haukaas, Rate adaptive video streaming over wireless networks explained and explored, MSc thesis, Department of Telematics, Norwegian University of Science and Technology, Trondheim, Norway, 2007.
[28] FFmpeg, 12 September 2010. <http://www.ffmpeg.org/>.
[29] Suramya Tomar, Converting video formats with FFmpeg, Linux Journal 2006 (146) (2006) 10.
[30] Ryad Ben-El-Kezadri, Farouk Kamoun, Guy Pujolle, XAV: a fast and flexible tracing framework for network simulation, in: Proceedings of the 11th International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Vancouver, British Columbia, Canada, October 27–31, 2008.
[31] TRAFIL download. <http://ru6.cti.gr/ru6/research_tools.php#TRAFL>.