simul_data
User's manual

Miguel Hernandez University [1]

May 3, 2011

Abstract

In this manual the installation and usage of the *simul_data* program is described, which is included within the *simul_xxx* set of programs. It has been developed by P. Pablo Garrido Abenza (pgarrido@umh.es), as a member of the Arquitectura y Tecnología de COMputadores Group (GATCOM) at the Miguel Hernandez University.

Contents

Chapter 1

Introduction

The aim of this tool is to extract data from several OPNET Modeler simulations results (.ov files), which have been generated by the *simul_run* program, which indeep invokes the *op_runsim* utility. The data exported are written to plain text files in order to be processed later. The program *simul_data* invokes to *ov2txt* for each simulation, although this program can be manually executed by the user (see below). Finally, this program can be executed in a cluster environment like Condor [2], executing several ov2txt instances in parallel, by only specifing the `-cluster` option.

Beside using the *simul_data* program, for doing this job it can also be used:

- The OPNET Modeler Graphical User Interface (GUI).
- From the command-line with the `op_cvov` tool.

Although using interactively the GUI is easier than other options, it has the disadvantage of working interactively when there are many simulations. When *simul_data* was developed, there wasn't other possibility appart from the OPNET application. However, currently (from OPNET v14.0?) there exists a command line utility called `op_cvov` for that purpose. Below are shown three examples of using that utility. In this first case, all the output files for all the scenarios in the project specified.

```
op_cvov -m simul_run                -mod_dirs ~/op_models/simul_run
    -output_file_path ~/op_models/simul_run/data/file.txt
    -vector_header -vector_data
```

In the second one, all output files for that scenario will be considered.

```
op_cvov -m simul_run-mymodel        -mod_dirs ~/op_models/simul_run
    -output_file_path ~/op_models/simul_run/data/file.txt
    -vector_header -vector_data
```

Finally, in the last example, all the vectorial data for the specific output file (.ov) will be exported. In any case, all the statistics will be exported to the .txt file.

```
op_cvov -m simul_run-mymodel-DES-1 -mod_dirs ~/op_models/simul_run
    -output_file_path ~/op_models/simul_run/data/file-1.txt
    -vector_header -vector_data
```

With *simul_data*, the first possibility is not available, that is, a scenario is allways needed to be specified, and the data to be exported can be from either one simulation, or all of them, or a range of simulations. Although more examples will be shown in section 3.4, in order to quickly compare both utilities, two uses of *simul_data* are shown in the following, "equivalent" to the second and third previous examples:

```
simul_data.sh -m simul_run-mymodel -i ~/op_models/simul_run/results
    -o ~/op_models/simul_run/data -f rh

simul_data.sh -m simul_run-mymodel -i ~/op_models/simul_run/results
    -o ~/op_models/simul_run/data -f rh -first 1 -last 1
```

In brief, the main differents between *simul_data* and `op_cvov` are:

1. With *simul_data* it is possible to select the statistics to be exported, instead of exporting all statistics contained in the .ov file. Moreover, such selected statistics can be specified in several ways (by index, by name, etc.).

2. The format of the output file generated by `op_cvov`, although it is plain text as *simul_data*, it is very difficult of processing automatically. For example, in case of extract data from several simulations (from one scenario or from a project), all the data will be written to the same file, whereas with *simul_data*a different text file will be generated for each .ov file, i.e. for each simulation.

3. With *simul_data* is possible to specify the output format. Although it is plain text with both tools, there are differences depending if the file will be processed with Excel or the R statistics package [1] (the end-of-file character, the separator character, the character used for decimal point in case of using Excel with a non-english language, etc.).

4. With *simul_data* it is possible to compute several functions with the vector data in order to get scalar data: MIN, MAX, AVG, MEAN, VAR, SUM, etc. Those operations can be defined with OPNET Modeler in order to generate output scalar data files (.os), except for the SUM function that is not offered by OPNET. The advantage of this possibility is that it is not necessary to generate the .os files for obtaining those values, as *simul_data* compute them from the vector data.

5. The *simul_data* program can execute in parallel the exporting process, running many *ov2txt* instances simultaneously, each one in a different processor if a Condor cluster [2] is used.

The set of programs *simul_xxx*, contains the following command-line tools:

- *simul_run*: used for program sets of simulations with OPNET Modeler without using the Graphical User Interface.
- *simul_data*: used for extracting the data obtained with the simulations. This program calls to *ov2txt* for each of the simulations run.
- *ov2txt*: invoked from the *simul_data* for extracting data from the binary vectorial files (.ov). It can also be manually invoked by the user.
- *simul_graphs*: used for generating graphs from the extracted data (R or gnuplot).
- *simul_reports*: used for generating a report containing the previously generated data and graph.

In addition to the aforementioned programs, there exists some other utilities specificallt designed when working with MANETs scenarios:

- *simul_geoloc*: used for generating network scenarios automatically, computing the partition grade according to the transmission range.
- *simul_ah2r*: used for converting the binary animation file (.ah) to a text format (.as).
- *simul_manet*: for generating easily .ef files with input parameters for the *simul_run* utility. For example, it is possible to specify a transmission range for the nodes, ant this program will write to the output file the appropriate values for the appropriate properties for some objects in the model.

On the contrary to the conventional MS-DOS/Unix script languages, these programs have been developed using the C programming language, so it has been easy to migrate them to other platforms, namely, Windows and Unix-like systems like Linux or Mac OS X[1].

[1]Currently OPNET Modeler is only available for Windows y Linux, although perhaps it could be executed on Mac OS X by using X.

Chapter 2

Installation

The instalation of the *simul_data* program is very similar to the other of the programs of the *simul_xxx* package. You only have to copy the binary to any directory and set some environment variables (`PATH` and `LD_LIBRARY_PATH`). In the following sections, how to set these environment variables is explained. In addition, the *ov2txt* program should be installed (see the following section).

2.1 Requeriments

The program *simul_data* requires that the program *ov2txt* be installed, which indeep requires to be compiled with the same version of the OPNET Modeler installed (see the *ov2txt* manual for more details). For convenience, the *simul_data* program does not invokes the executable directly but a script named *ov2txt.sh* (Linux) or *ov2txt.bat* (Windows), so, it should exists in an accessible path. That script executes the appropriate version of the *ov2txt* program according to the versión of the OPNET Modeler.

2.2 Windows

Let us suppose that the directory choosen for copying the *simul_data* program is:

```
C:\Program files\utils
```

It is recommended to modify the `PATH` environment path. This can be done as explained in the following:

- Windows 98 or earlier: edit the C:\AUTOEXEC.BAT file, adding the following line at the end. It would be possible that the path needs to be enclosed within quotes if it is more than 8 characters length or it contains blanck spaces:

  ```
  SET PATH=%PATH%;C:$\Program files\utils
  ```

- Windows NT, 2000, XP, ...: within the Control Panel, choose System > Advanced > press the button [Environment variables]. Then the path could be

added to the `PATH` environment variable, separated by a semicolon ';'. Alternatively, the Autoexec.NT file can be modified in a similar way as explained for the Autoexec.bat file.

2.3 Linux

Let us suppose that the directory choosen for copying the *simul_data* program is:

```
$HOME/bin
```

Under Linux it is necessary to set the `PATH` and `LD_LIBRARY_PATH` environment variables. In the later we should to add the path for the OPNET libraries, as it is necessary in order to run programs compiled with the API EMA (built with op_mkema). This path depends on the home directory for OPNET and version. The way to do that depends on the shell or command-line interpreter used (bash, ksh, ...):

- Linux (shells: bash, ksh, zsh, sh): $HOME/.bash_profile or $HOME/.profile:

```
PATH=.:$PATH:$HOME/bin
LD_LIBRARY_PATH=/usr/opnet/16.0.A/sys/pc_intel_linux/lib
export PATH
export LD_LIBRARY_PATH
```

- Linux (shells: csh, tcsh): $HOME/.login:

```
setenv PATH .:$PATH:$HOME/bin
setenv LD_LIBRARY_PATH /usr/opnet/16.0.A/sys/pc_intel_linux/lib
```

Some Linux distributions have problems with the `LD_LIBRARY_PATH`, and user cannot set the value. This is why it is necessary to execute that each time that either you are going to execute the program or you open the console.

In order to simplify the process, you can write a little script called ov2txt.sh similar to the following, and put it together to the binary. In order to pass the arguments from the script to the executable, you can use `"$@"` (with the quotes) instead of `$*` too.

```
#!/bin/sh
export LD_LIBRARY_PATH=/usr/opnet/16.0.A/sys/pc_intel_linux/lib
exec simul_data-en-x86.bin $*
```

Chapter 3

Usage of the program

In this chapter, the usage of the *simul_data* program is explained. It is supposed that we have written the *simul_data*.sh script explained in the previous chapter.

3.1 Syntax

```
simul_data -?
simul_data -m <input-file> [-w <path>] [-i <path>] [-o <path>]
           [-q] [-v] [-f <output-format>] [-x <data-to-export>] [-s <suffix>]
           [-first <n>] [-last <n>] [-dont_run]
           [-cluster <np> <met> [<notif> <email>]]
```

3.2 Description

The *simul_data* program extract data from the binary .ov files generated as a result of the run of a OPNET Modeler simulations sequence, exporting the data to plain text files in order to be processed lately with Excel, the R statistical package, etc. These text files have a tabular structure, with columns separated with some separator character (csv). The program can be executed either sequential or in parallel with a Condor cluster.

The program does not run interactively. Instead, the program begins to extract data immediately upon invocation and goes on until the process finishes, without interruption. Because of that, it is necessary to specify some options when invoking the program. These are the options to *simul_data*:

```
Overall arguments (help, ...):
-------------------------------------
   -?: shows this help.

   -v: verbose mode.

Arguments to pass through to 'ov2txt':
-------------------------------------
The following options will be passed through directly to ov2txt.
```

```
Please, see the ov2txt help for more details:
   -m, -w, -i, -o, -q, -v, -f, -x, -s

Arguments for defining the sequence of simulations:
---------------------------------------------------
If the model spefified with the -m option is 'mymodel', the files
used will be::
   'mymodel-DES-001.ov', 'mymodel-DES-002.ov', ...

By default, the first file is always -001, and the process will end
when a file is not found, unless the -first or -last be found:
   -first <n>: index of the first simulation for begin the process.
        By default, it is the first one (1).
   -last  <n>: index of the last simulation for ending the process.
        By default, the last file found when incrementing the index.

   -dont_run: used for displaying only the information for data to be
        extracted, generating the necessary files to launch jobs
        if a cluster Condor is used.

Arguments for using a cluster (Condor):
---------------------------------------
   -cluster <np> <met> [<notif> <email>]: run the specified simulation set
        in parallel (cluster Condor). If 'simul_data' is executed in a cluster
        and this option is not specified, the simulations (jobs) will be
        executed sequentially in the same processor.
        The arguments for this option are:
           <np>     = number of processors to be used, that is, maximun number
                      of simultaneous jobs, one per simulation (np>=0).
                      Since OPNET Modeler users must own a number of valid
                      licenses, the <np> value should be less or equal than the
                      available licenses; otherwise, many simulations will fail.
           <met>    = method used to launch the necessary jobs in Condor:
                      1-independent jobs; 2-DAG; 3-Parametrized DAG.
                      In case the number of simulations (jobs) be greater than
                      <np>, method 1 can not be used, as it could mean that.
                      there is not enough OPNET licenses for them.
                      If that case is detected, method 2 will be used.
           <notif>  = events to notify to the indicated <email>:
                      0-Never; 1-Completed; 2-Errors; 3-Always
           <e-mail> = e-mail address to send the notifications.
```

3.3 Files

The *simul_data* program use the binary .ov files as input (`mymodel-DES-001.ov`, `mymodel-DES-002.ov`, ...), which have been written by the `simul_run` tool (and `op_runsim`). On the other hand, the program writes a text file for each .ov file as output (with the .data or .xls suffix).

Optionally, if a cluster Condor is used (by specifying the `-cluster` option), the program *simul_data* will create automatically all the necessary files to launch jobs for running the simulations in parallel (.sh/.job/.dag), and the execution of those jobs will write three more output files with messages and errors (.out/.err/.log). The later will be written in a subdirectory below the current directory (or working direc-

tory), named: `./condor1`, `./condor2`, or `./condor3`, depending on the method used to launch jobs (1-3). For more details about the generated files when using the cluster Condor see the *simul_run* manual.

3.4 Examples

The following examples show how to use the *simul_data* program. In brief, the same examples shown in the *ov2txt* manual could be applied here, as the most of the arguments passed from the command-line are the same. Only four arguments are new to *simul_data*: `-first`, `-last`, `-dont_run`, and `-cluster`. As we can see, all of them are related to run the *ov2txt* several times (or none), in order to extract the data for each simulation run within a simulations sequence, and to specify whether to use or not use a cluster Condor. The remainder arguments (`-m`, `-w`, `-i`, `-o`, `-q`, `-v`, `-f`, `-x`, `-s`) will be passed to *ov2txt* in each invocation. For more details and examples about those options, see the *ov2txt* manual.

This first example extract the data for all the simulations run using the model `mymodel.nt.m`. The results for those simulations, that is, the .ov files generated by *simul_run* (in fact, by *op_runsim*), were stored in a relative path to the current directory named `./results`. Please, note that the .ov suffix should not be specified at the end, because we are not specifying the file name but the corresponding name model. By default, the final .ov files used is determined by searching throught the paths listed in the `mod_dirs` preference within OPNET Modeler, exactly in that order. However, you can force the path to be used by using the `-i` option, without needing to modify such preference.

```
simul_data -m mymodel  -i ./results  -x 1,2,3  -f rh
```

In the previous example, the `-first 1` argument is assumed, that is, the process will start by extracting data from a file named `mymodel-DES-001.ov`. On the other hand, as the `-last` option is not specified, the process will go on till the last `mymodel-DES-xxx.ov` be found, where xxx=001,002,... In each case, only the first three statistics stored will be exported, and the format used for the output file will be suitable for the R package and a header line will be included at the beginning of the file with the names of those statatistics.

If we need to extract data only for a range or a specific simulation, we can either run the *ov2txt* utility manually, or *simul_data* specifying the `-first` and `-last` arguments, as shown in the following example:

```
simul_data -m mymodel  -i ./results  -x 1,2,3  -f rh
           -first 5 -last 6
```

In this way, only the `mymodel-DES-005.ov` and `mymodel-DES-006.ov` files will be used. Alternatively, as it has been said previously, we could use the *ov2txt* program, executing it twice in this case:

```
ov2txt -m mymodel-DES-005  -i ./results  -x 1,2,3  -f rh
ov2txt -m mymodel-DES-006  -i ./results  -x 1,2,3  -f rh
```

In case there would be a great number of simulations, it is recommended to run the program in parallel if a cluster Condor is available with multiple processors. The program *simul_data* allows to make use of the cluster easily only by appending the option `-cluster` with the appropiate arguments. So, the user do not need the knowledge needed to launch jobs in such cluster. The arguments for this option are the number of processors (i.e. the maximum number of *ov2txt* instances running simultaneously), and the method used to launch jobs (see the *simul_run* manual for more details).

```
simul_data -m mymodel  -i ./results  -x 1,2,3  -f rh
           -first 1 -last 500  -cluster 100 3
```

In that example, 500 .ov files will not be extracted in a sequential way but using the cluster for running till 100 *ov2txt* instances simultaneously (or till the number of availbale processors). The user does not need to know anything about how to launch jobs in the cluster as all the necessary files (.job/.dag/.sh/...) are generated automatically in a subdirectory named: `./condor1`, `./condor2`, or `./condor3`, depending on the method used to launch jobs (1-3).

Finally, it could be useful the `-dont_run` option, specially when using a cluster for the first time. In that case, the files for the Condor cluster will be generated but do not run, allowing the user to check them before running, or even modify them and running manually without using the *simul_data* program.

Bibliography

[1] R Development Core Team. R: A language and environment for statistical computing. ISBN 3-900051-07-0, 2007.

[2] The Condor Team. Condor.