ClusterGUI, an Application to Launch OPNET Simulations within Resource Managed Environments

Carlos Núñez Castillo, Gonzalo Zarza, Diego Lugones, Javier Navarro, Daniel Franco, Emilio Luque
Computer Architecture and Operating Systems Department,
Universitat Autònoma de Barcelona, Spain.
E-mail: {carlos.nunez, gonzalo.zarza, diego.lugones, javier.navarro}@caos.uab.es, {daniel.franco, emilio.luque}@uab.es

Abstract
This work presents ClusterGUI; an application specialized in the submission and control of OPNET simulation jobs to the Oracle Grid Engine (previously known as Sun Grid Engine - SGE) Resource Manager, using the Distributed Resource Management Application API (DRMAA) libraries. With a Resource Manager acting as a middleware between cluster resources and simulation runs, traditional tools cannot be directly used to launch the simulations. With ClusterGUI we can send multiple simulations to a resource manager in a cluster in order to fully use available resources. We compare simulations results by launching many standalone runs against a resource-managed distributed approach, to perform parametric studies more quickly by launching simulations concurrently.

Introduction
Simulation has become an important technique in the development of many areas of science and engineering, such as communication networks, distributed systems, scientific computation or research and development in general, among others.

In order to compare systems performance analysis under different scenarios or between design alternatives, modeling and simulations techniques provides an efficient environment [1].

Computational resources available to perform complex simulations are constantly increasing. High Performance Computing (HPC) clusters and computer farms require the use of specialized resources administration techniques. These techniques are necessary because the manual assignment of several hundreds of jobs into its proper resources would become unmanageable complex.

OPNET modeler [2] allows the executions of simulations in a distributed environment. To accomplish this task, the modeler assumes the whole computing environment is dedicated exclusively to this task. If others applications try to use those resources as well, then conflicts arises. Some conflicts can be caused by different concurrent executions. For example an application can interfere with other active process that demands all available resources exclusively. By allowing these two executions to run simultaneously, one of them would be severely affected in its results. Other important issue is the way resources are assigned. As computational resources in HPC systems are generally large, resources assignment should be done automatically.

In order to use those resources accordingly and in an automated manner, specific strategies must be taken into account. Here, resource management tools must be used, such as Oracle Grid Engine [3]. This tool takes care of executing jobs and distributes them across all available computing resources.

The main goal of this paper is to introduce a tool used as a frontend between a resource managed computing environment and OPNET simulations.

This paper is organized as follows. In section II we give some background about OPNET and Distributed Resource Management Systems. In section III, we describe our ClusterGUI Architecture and in section IV we give some ClusterGUI application descriptions. In section V, we show a basic usage guide of our tool. Concluding remarks are offered in section VI.

Background
This section gives a general overview of the OPNET modeler tool, Oracle Grid Engine resource manager, and the API used to interconnect these two architectures.

OPNET Modeler
Opnet provides a Discrete Event Simulator (DES) engine and offers a hierarchical modeling environment with an enhanced C++ language. This suitable environment allows defining network components behavior by a Finite State Machine approach (FSM), and it supports detailed specification of protocols, resources, applications, algorithms, and queuing policies.
Simulations can be executed in three different ways:

- From the OPNET GUI, using either the Project Editor or the Simulation Sequence Editor.
- From the command line, using op_runsim.
- Creating an executable file, with op_mksim, and run it from the command line or Simulation sequence Editor.

Decision about which alternative to use are based on specific user considerations, such as complex development, maximum availability of computing resources, debugging among others. OPNET GUI is the most used option. However, there are specific situations where the other two alternatives are preferred because of the options they provide. Also, the flexibility offered by the alternatives to the GUI makes them the right choice within complex environments, such as those encountered in HPC.

OPNET Architecture
A typical OPNET project contains basically models files (network, nodes, process, etc), files generated with the

configuration for each specific execution (such as Environment Files), as well as results files (such as Output Vector files).

In this document, we will use CSMA/CD Official OPNET tutorial project. The network domain used on the tutorial is shown on Fig. 1. As extracted from the project documentation, the goal of the CSMA/CD project is to observe how the performance of the protocols varies as a function of channel traffic. The interarrival time input parameter was varied in a series of simulations to produce different levels of traffic and, therefore, different levels of throughput. Twelve simulations have been run, each with a different interarrival time value, as seen on Fig. 2. Each of these simulations has its own particular configuration, which will be used as input parameters by the simulation engine.

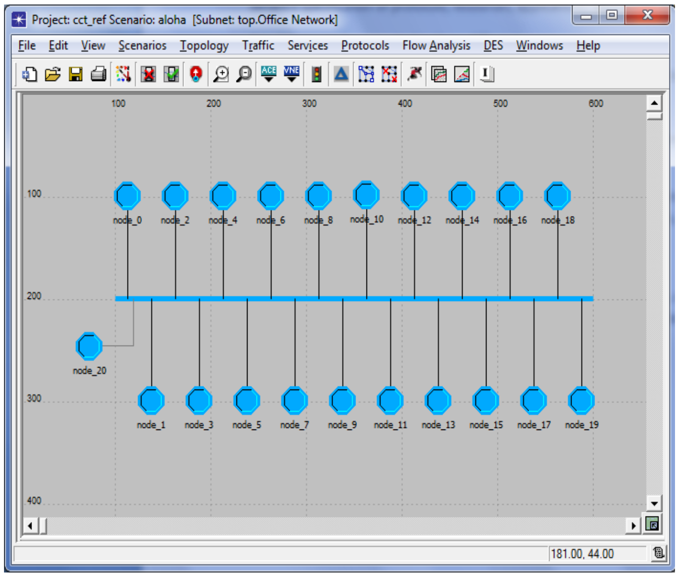
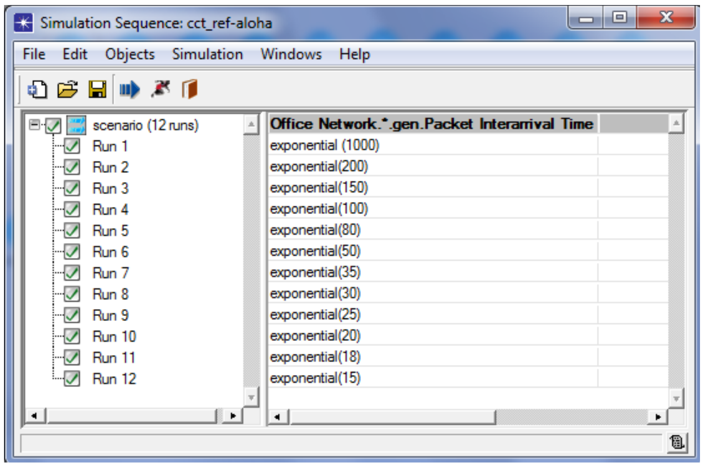

Project: cct_ref Scenario: aloha [Subnet: top.Office Network]
File Edit View Scenarios Topology Traffic Services Protocols Flow Analysis DES Windows Help
node_0
node_2
node_4
node_6
node_8
node_10
node_12
node_14
node_16
node_18
node_20
node_1
node_3
node_5
node_7
node_9
node_11
node_13
node_15
node_17
node_19
181.00, 44.00

Fig. 1. CSMA/CD OPNET Tutorial Project.


Simulation Sequence: cct_ref-aloha
File Edit Objects Simulation Windows Help

scenario (12 runs)	Office Network.*.gen.Packet Interarrival Time
Run 1	exponential (1000)
Run 2	exponential(200)
Run 3	exponential(150)
Run 4	exponential(100)
Run 5	exponential(80)
Run 6	exponential(50)
Run 7	exponential(35)
Run 8	exponential(30)
Run 9	exponential(25)
Run 10	exponential(20)
Run 11	exponential(18)
Run 12	exponential(15)

Fig. 2. CSMA/CD Simulation Sequence Editor.

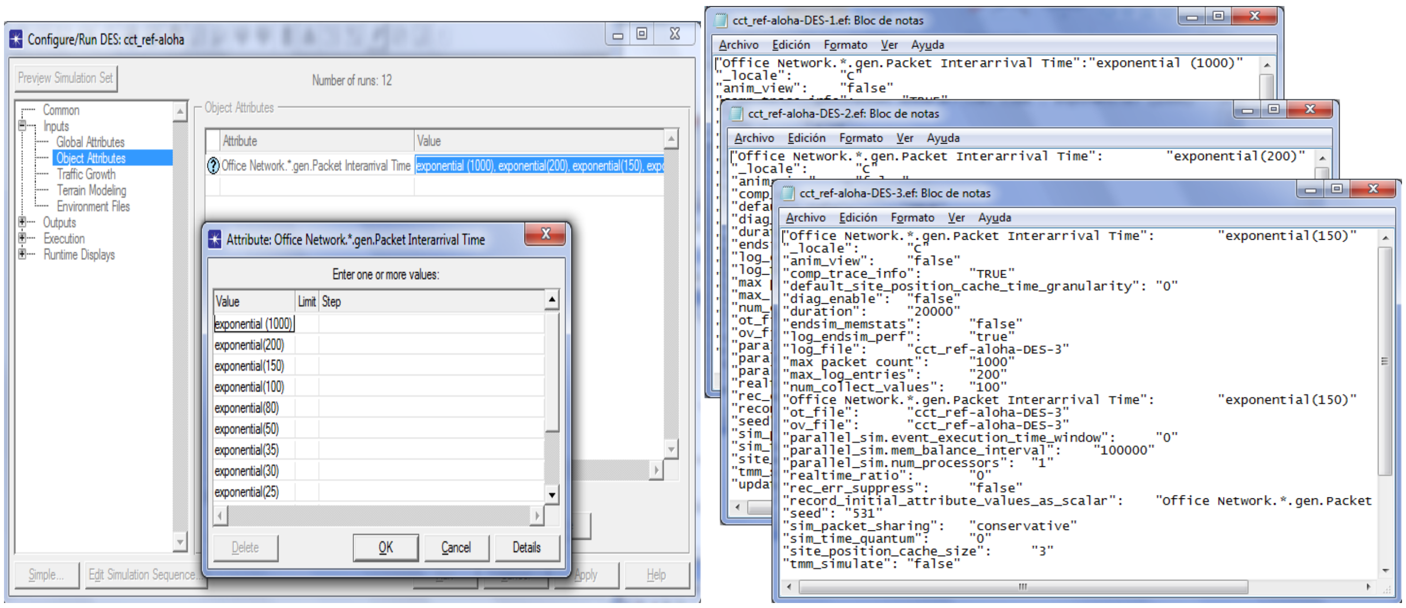
Environment Files

As mentioned before, one simulation has its own configuration parameters. OPNET then creates particular files, called environment Files identified with file extension ".ef", for each execution in the sequence. As an example, Fig. 3. depicts a basic relation between input parameters specified in the Project Editor and the files created it the file system.

We can see from the example that each configuration parameter "Packet Interarrival Time" (exponential (1000), exponential (200), exponential (150)) is set in each corresponding "Environment File" created. All these files will be later referenced at the ClusterGUI application, in order to identify particular simulations runs that will be sent to the queue manager as a regular job.

Oracle Grid Engine (SGE) Architecture

Oracle Grid Engine[3] is a Distributed Resource Management platform, basically in charge of queuing and scheduling jobs, matching jobs to most suitable execution hosts, managing resources such as licenses and applying resources allocation polices. It allows the use of a shared infrastructure by many



Attribute	Value
Office Network.*.gen.Packet Interarrival Time	exponential (1000), exponential(200), exponential(150), exp

Value	Limit	Step
exponential (1000)		
exponential(200)		
exponential(150)		
exponential(100)		
exponential(80)		
exponential(50)		
exponential(35)		
exponential(30)		
exponential(25)		

cct_ref-aloha-DES-1.ef:

```
"Office Network.*.gen.Packet Interarrival Time":"exponential (1000)"
"_locale":        "C"
"anim_view":      "false"
```

cct_ref-aloha-DES-2.ef:

```
"Office Network.*.gen.Packet Interarrival Time":        "exponential(200)"
"_locale":        "C"
```

cct_ref-aloha-DES-3.ef:

```
"Office Network.*.gen.Packet Interarrival Time":        "exponential(150)"
"_locale":        "C"
"anim_view":      "false"
"comp_trace_info":        "TRUE"
"default_site_position_cache_time_granularity": "0"
"diag_enable":  "false"
"duration":     "20000"
"endsim_memstats":        "false"
"log_endsim_perf":        "true"
"log_file":     "cct_ref-aloha-DES-3"
"max packet count":       "1000"
"max_log_entries":        "200"
"num_collect_values":     "100"
"Office Network.*.gen.Packet Interarrival Time":        "exponential(150)"
"ot_file":      "cct_ref-aloha-DES-3"
"ov_file":      "cct_ref-aloha-DES-3"
"parallel_sim.event_execution_time_window":     "0"
"parallel_sim.mem_balance_interval":    "100000"
"parallel_sim.num_processors":  "1"
"realtime_ratio":         "0"
"rec_err_suppress":       "false"
"record_initial_attribute_values_as_scalar":    "Office Network.*.gen.Packet
"seed": "531"
"sim_packet_sharing":     "conservative"
"sim_time_quantum":       "0"
"site_position_cache_size":       "3"
"tmm_simulate": "false"
```

Fig. 3. Input Parameters. Modeler and corresponding files.

users/jobs. It is typically used on a computer farm or High Performance Computing (HPC) cluster. Fig. 4 depicts a basic environment where resources are managed by a distributed resource manager. Here, there is a server where the software in charge of managing resources is installed. This server receives user's jobs request, and responses from the system when necessary. After receiving a job, this server dispatches it to the proper computing resources. Resources in this case could be anything installed in the cluster. In this example, resources are a blade server, a main frame or a set of racks of computing nodes linked together by a local area network.

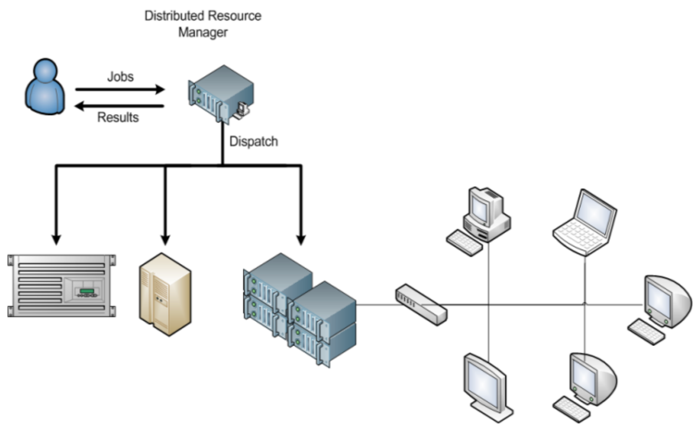


Fig. 4. Resource Management.

There are two main alternatives available to make resources available to a particular job through SGE interface. One is the use of a command line approach, by using directives that explicitly work with jobs, such as the task of submitting, logging or managing these jobs. These directives are qsub, qmon, etc. The other approach is to use some specialized programming libraries, such as DRMAA, in order to obtain some degree of process automation or customization. Both approaches start execution daemons which are properly managed by the Qmaster/Scheduler, as shown in Fig. 5.

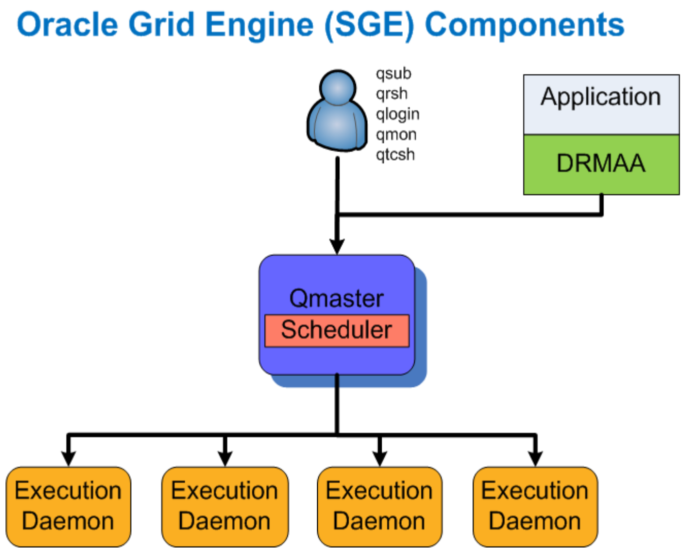


Fig. 5. SGE Architecture.

DRMAA Architecture

DRMAA [4] stands for Distributed Resource Management Application API, a set of libraries for the submission and control of jobs to one or more Distributed Resource Management (DRM) systems. Different API Implementation exists today, such as Oracle Grid Engine with DRMAA C, Condor with DRMAA C, GridWaty with DRMAA C/Java among other. Fig. 6 shows DRMAA general architecture.

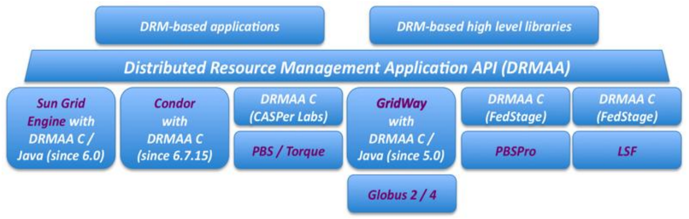

DRM-based applications
DRM-based high level libraries
Distributed Resource Management Application API (DRMAA)
Sun Grid Engine with DRMAA C / Java (since 6.0)
Condor with DRMAA C (since 6.7.15)
DRMAA C (CASPer Labs)
PBS / Torque
GridWay with DRMAA C / Java (since 5.0)
Globus 2 / 4
DRMAA C (FedStage)
PBSPro
DRMAA C (FedStage)
LSF

Fig. 6. DRMAA Architecture

Distributed Simulations with OPNET

OPNET [2] allows multiple discrete event simulations to be executed at the same time using one or more computers. This Modeler feature tries to start a specified number of simulations on each specified computer, and as each job is finished, Modeler tries to send another job to be executed. This process continues until all simulations waiting to be executed have been run. As stated before, the main limitation of this approach is that OPNET assumes that all resources are dedicated to its simulations. On the contrary, in mostly all HPC environments the configuration is shared. Here, there are many simultaneous simulations trying to access those resources. Generally, HPC installations have a resource manager installed. This manager is responsible for accepting, scheduling, dispatching, and managing the remote and distributed execution of large numbers of standalone, parallel or interactive user jobs. Also, licenses control could make a job to finish beforehand. This happens when a job is assigned to a computing node, and OPNET then makes proper licenses control and no licenses available for execution are found. This forces a submitted and already assigned job to exit without being executed.

ClusterGUI Architecture

ClusterGUI application is designed with integration in mind. ClusterGUI main goal is to become an interface between OPNET modeler and the Distributed Resource Management installed. With this tool all OPNET simulations can be safely sent to a shared cluster. In cooperation with the DRM, ClusterGUI can monitor and manage all running jobs, giving the user full flexibility in order to control simulations.

ClusterGUI application is based on the **Oracle Grid Engine** [3] **with DRMAA Java API.** As being based on the Java API [5], [6]**,** this implementation allows a high degree of inter-platform portability. ClusterGUI provides a front end to OPNET users in order to launch simulations jobs. Recall that all OPNET simulations must be modeled/tested within OPNET Modeler

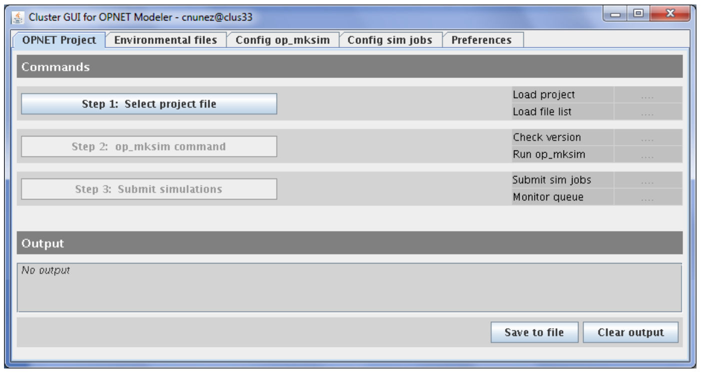
GUI interface first. ClusterGUI does not provide a GUI interface to perform these activities.

ClusterGUI follows the defined OPNET standard to run simulations. It can execute the executable program generated by op_mksim, and also can use the program op_runsim in order to launch simulations. op_runsim and simulation executables created by op_mksim work similarly. The only difference is that op_runsim dynamically rebuilds the network repository and op_mksim will link static library information into the simulation executable. ClusterGUI basically perform five basic steps. In the first step it opens and loads the OPNET project file. Then, the second step begins and ClusterGUI loads all the configuration files available to this project. These are the "environmental files" as showed in Fig. 3. Once all the environmental files are correctly loaded, the user can select which ones will be executed. The third step is related to the process of generating the proper executable simulation files. In order to perform this step the program op_mksim is invoked with the correct input parameters. If no errors are found after the execution of op_mksim, then proper executable sim files are created in the output directory. Step four could then be performed. This step allows ClusterGUI to submit simulations to the DRM as regular jobs. When all simulations are already sent to the DRM´s queue by ClusterGUI, then the monitoring process starts. This corresponds to the fifth and last step of ClusterGUI application execution. While in this fifth step, an application listener is active and regularly collects status information about running jobs. Also, complete control of sent jobs is achieved. Any job can be stopped or killed from ClusterGUI main screen at any time.

These steps explained above allow the execution of many simultaneous simulations to be performed, under the presence of a DRM system. All results are collected properly. The directory containing the model being simulated must be shared by the local and remote computers. The output (.ov) files produced by the simulation are created in the shared model directory. OPNET requirements, such as licenses required based on the architecture/platform used, are still maintained.

ClusterGUI Description

Cluster GUI main application screen is shown in Fig. 7. From this screen all the simulations options can be modified/verified before sending a job to the queue manager. It is basically composed of a number of tabs, specifically distributed in order to show related information in an ordered manner. Each of the tabs will be described below.



No output

Fig. 7. Main Screen.

OPNET Project Tab

Basically, this tab allows the execution of simulations. Three steps are at minimum required in order to send jobs to the DRM. These steps are explained here:

1) The first step allows the user to select the project to be loaded.
2) The second step, allows the execution of the op_mksim program, which is a utility that binds a simulation together from its components: the simulation object code file, the associated model archive, and the Simulation Kernel libraries.
3) Once all the previous steps have been performed correctly, then the simulations are ready to be sent to the DRM as regular jobs in the third step.

This tab has an output log viewer, where all logs from the application itself as well as the simulations will be printed. Beside action buttons, there are hints texts which indicate each task final status.

Environmental files Tab

This tab shows all the environmental files available from this project. Environmental files contain simulation execution parameters, such as duration, output files, seed, etc. These files are created for every run generated during parametric simulations, while in OPNET editor. Here, each file has a contiguous check box available, in order to decide whether it is to be included or not for submission to the DRM. File names corresponds to different runs available in the selected project. Double click in each file will show its contents.

Config op_mk_sim Tab

The purpose of op_mksim is to bind (link) together process model object code files and the simulation kernel into an executable simulation program. The simulation repository associated with the indicated network model is always reconstructed before it binds the simulation. This tab permits changes to the default preferences of op_mksim. Default preferences are Standard, Diagnostics, Development and Licensing, according to official OPNET documentation.

This tab is useful when some particular features must be changed, such as for example:

- optional binder flags used when making a static 32-bit executable
- libraries appended to binder command when making a static 64-bit executable
- binder program used to create executables
- optional flags passed when compiling for a 32/64-bit kernel
- C/C++ compiler program

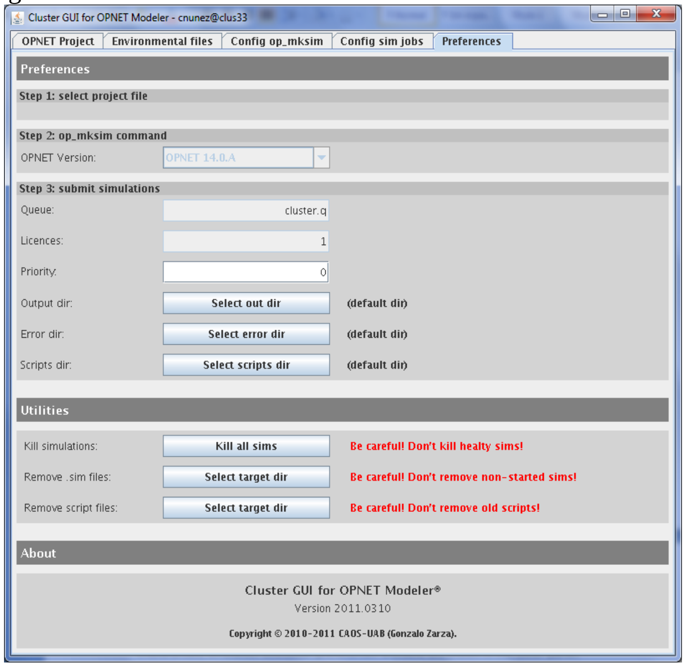
The lower panel outputs op_mksim –help command, showing available preferences that can be changed.

Configure sim jobs Tab

Once the op_mksim command is executed successfully, then a new sim job file is created. Simulation execution preferences available are shown in the lower panel.

Preferences Tab

Available ClusterGUI preferences are managed in this tab. Output directories can be set, also information about the queue manager and opnet licenses available are shown. Some utilities to manage .sim files job can be started from this tab, as shown in Fig. 8.



Preferences

Step 1: select project file

Step 2: op_mksim command

OPNET Version: OPNET 14.0.A

Step 3: submit simulations

Queue: cluster.q

Licences: 1

Priority: 0

Output dir: (default dir)

Error dir: (default dir)

Scripts dir: (default dir)

Utilities

Kill simulations: Be careful! Don't kill healty sims!

Remove .sim files: Be careful! Don't remove non-started sims!

Remove script files: Be careful! Don't remove old scripts!

About

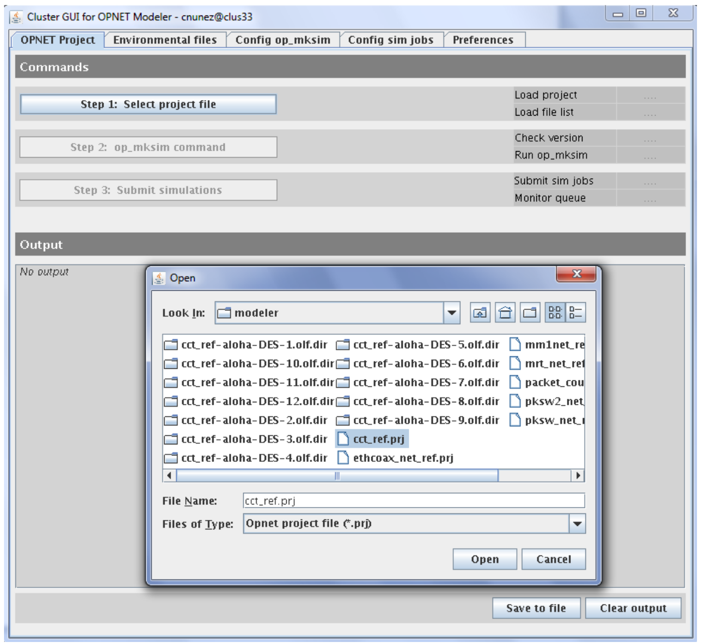
Cluster GUI for OPNET Modeler®
Version 2011.0310
Copyright © 2010-2011 CAOS-UAB (Gonzalo Zarza).

Fig. 8. Preferences Tab.

ClusterGUI Basic Usage

In order to use ClusterGUI to send OPNET simulations to a DRM, a few easy steps are needed, which are mentioned here. We will use CSMA/CD Official OPNET tutorial project.

1) Launch the ClusterGUI application.
2) Click on "Step 1 Select Project File" to browse the .prj file in the file system, as shown in Fig. 9. Once loaded, the output log shows a summary of what is available in this project.



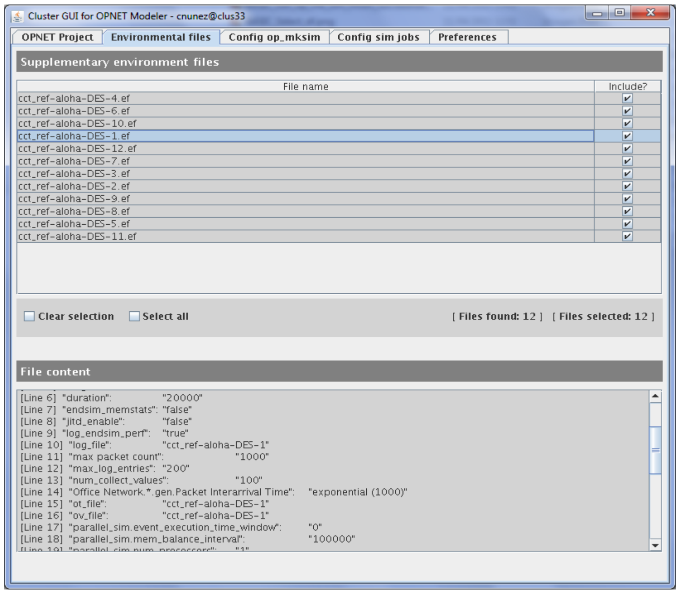
cct_ref-aloha-DES-1.olf.dir
cct_ref-aloha-DES-10.olf.dir
cct_ref-aloha-DES-11.olf.dir
cct_ref-aloha-DES-12.olf.dir
cct_ref-aloha-DES-2.olf.dir
cct_ref-aloha-DES-3.olf.dir
cct_ref-aloha-DES-4.olf.dir
cct_ref-aloha-DES-5.olf.dir
cct_ref-aloha-DES-6.olf.dir
cct_ref-aloha-DES-7.olf.dir
cct_ref-aloha-DES-8.olf.dir
cct_ref-aloha-DES-9.olf.dir
cct_ref.prj
ethcoax_net_ref.prj
mm1net_re
mrt_net_ref
packet_cou
pksw2_net
pksw_net_

File Name: cct_ref.prj

Files of Type: Opnet project file (*.prj)

Fig. 9. Browse and Open a Project.

3) Change to "Environmental files" Tab and select the proper .ef files to be included, as shown in Fig. 10.



Supplementary environment files

File name	Include?
cct_ref-aloha-DES-4.ef	✓
cct_ref-aloha-DES-6.ef	✓
cct_ref-aloha-DES-10.ef	✓
cct_ref-aloha-DES-1.ef	✓
cct_ref-aloha-DES-12.ef	✓
cct_ref-aloha-DES-7.ef	✓
cct_ref-aloha-DES-3.ef	✓
cct_ref-aloha-DES-2.ef	✓
cct_ref-aloha-DES-9.ef	✓
cct_ref-aloha-DES-8.ef	✓
cct_ref-aloha-DES-5.ef	✓
cct_ref-aloha-DES-11.ef	✓

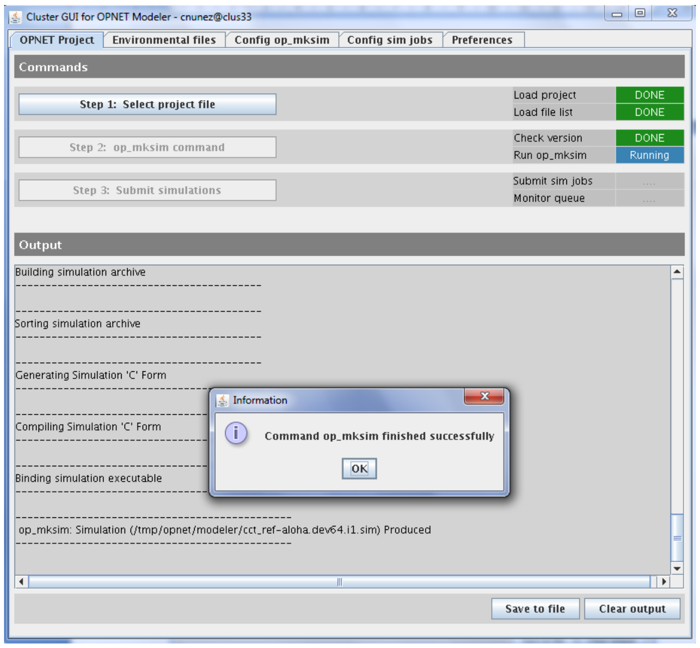
[Files found: 12] [Files selected: 12]

File content

```
[Line 6]  "duration":            "20000"
[Line 7]  "endsim_memstats": "false"
[Line 8]  "jitd_enable":         "false"
[Line 9]  "log_endsim_perf":   "true"
[Line 10]  "log_file":            "cct_ref-aloha-DES-1"
[Line 11]  "max packet count":                  "1000"
[Line 12]  "max_log_entries":  "200"
[Line 13]  "num_collect_values":                "100"
[Line 14]  "Office Network.*.gen.Packet Interarrival Time":    "exponential (1000)"
[Line 15]  "ot_file":             "cct_ref-aloha-DES-1"
[Line 16]  "ov_file":             "cct_ref-aloha-DES-1"
[Line 17]  "parallel_sim.event_execution_time_window":       "0"
[Line 18]  "parallel_sim.mem_balance_interval":              "100000"
```

Fig. 10. Select files to be included in this run.

4) Back into the main screen, click on "Step 2: op_mksim_command" to start the compilation process. This should take some time according to the selected .ef files from previous step. Assuming there were no errors during the op_mk_sim process, an information box appears and it notifies that the process ended correctly, as shown in Fig. 11. Now, proper simulation files (.sim) are created and they are ready to be sent to the DRM.

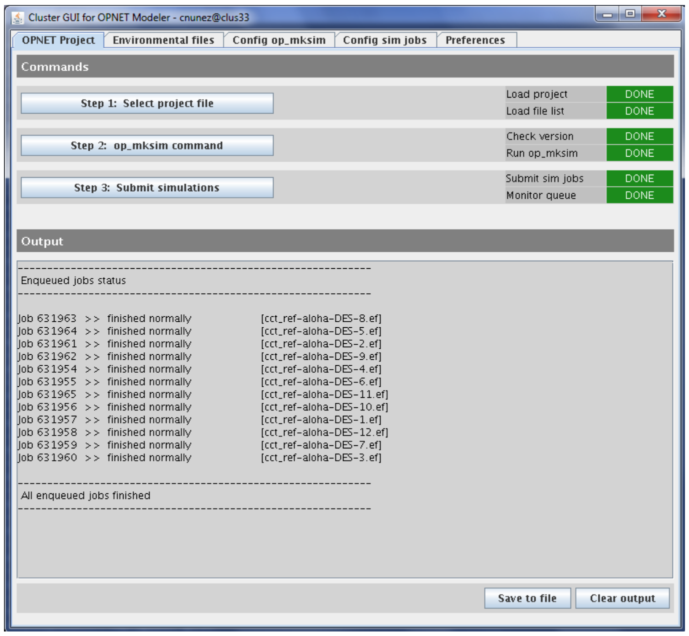


```
Building simulation archive
----------------------------------------

----------------------------------------
Sorting simulation archive
----------------------------------------

----------------------------------------
Generating Simulation 'C' Form

----------------------------------------
Compiling Simulation 'C' Form

----------------------------------------
Binding simulation executable

----------------------------------------
op_mksim: Simulation (/tmp/opnet/modeler/cct_ref-aloha.dev64.i1.sim) Produced
----------------------------------------
```

```
Command op_mksim finished successfully
```

Fig. 11. Running op_mk_sim command.

5) Click on "Step 3: Submit Simulations" to effectively send the .sim files to the DRM. The output log should show the sending job status. Once a job is correctly sent, the ClusterGUI application can be safely closed. Jobs are now waiting for resources and its execution is ruled by DRM policies, as shown in Fig. 12.



```
----------------------------------------------------------
Enqueued jobs status
----------------------------------------------------------

Job 631963  >>  finished normally          [cct_ref-aloha-DES-8.ef]
Job 631964  >>  finished normally          [cct_ref-aloha-DES-5.ef]
Job 631961  >>  finished normally          [cct_ref-aloha-DES-2.ef]
Job 631962  >>  finished normally          [cct_ref-aloha-DES-9.ef]
Job 631954  >>  finished normally          [cct_ref-aloha-DES-4.ef]
Job 631955  >>  finished normally          [cct_ref-aloha-DES-6.ef]
Job 631965  >>  finished normally          [cct_ref-aloha-DES-11.ef]
Job 631956  >>  finished normally          [cct_ref-aloha-DES-10.ef]
Job 631957  >>  finished normally          [cct_ref-aloha-DES-1.ef]
Job 631958  >>  finished normally          [cct_ref-aloha-DES-12.ef]
Job 631959  >>  finished normally          [cct_ref-aloha-DES-7.ef]
Job 631960  >>  finished normally          [cct_ref-aloha-DES-3.ef]

----------------------------------------------------------
All enqueued jobs finished
----------------------------------------------------------
```

Fig. 12. Submit a Simulation.

When a job finishes its execution, ClusterGUI application writes two output files in the home directory (or where output dir preferences were set). One file contains the DRM errors, if any, and the second contains the output of the simulation. Both files differ only in a suffix. They are identified by the name of the job + "extension" + PID of the job. "Extension" could be ".e" for the former (error) and ".o" for the latter (output).

6) When all simulations ends and no errors were produced, the output vector files will be generated properly, according to the model settings. Now, we can go back to OPNET modeler to visualize our results, as if they were executed from this GUI, as shown in Fig. 13.

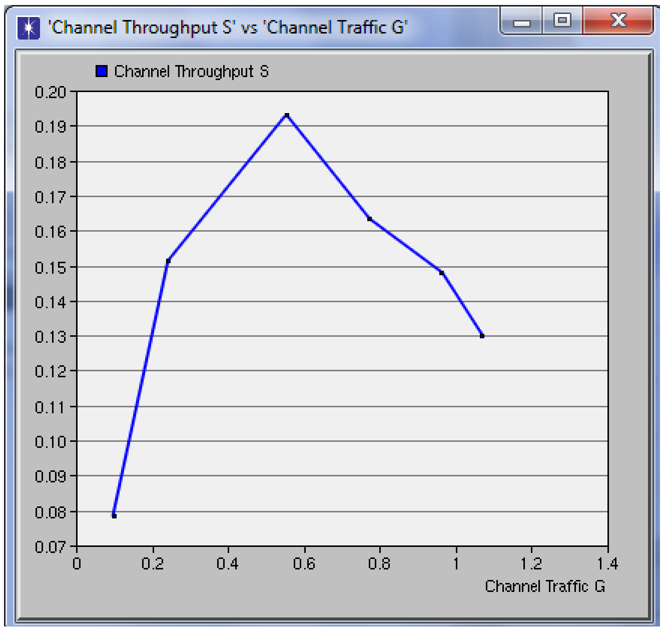
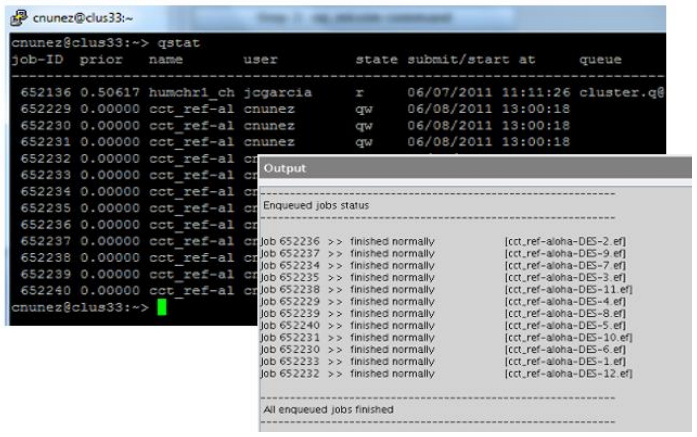


Fig. 13. Results Browser in OPNET GUI.

Jobs Control

ClusterGUI continuously monitors the status of all simulations jobs executed via its GUI. A DRM manages jobs from different sources, and ClusterGUI filters this information to show only those jobs started from the application. A typical output of ***qstat,*** the command line program to show cluster queue summary in SGE, is shown in Fig. 14. From the console at this figure, we can see that other user is also using the cluster. The DRM has queued our jobs until all resources demanded, including OPNET licenses, are available. This figure also depicts ClusterGUI output log. Here, job statuses are indicated for each job. In this example, all simulations finished normally.



```
cnunez@clus33:~> qstat
job-ID  prior   name       user         state submit/start at     queue
-----------------------------------------------------------------------------
 652136 0.50617 humchr1_ch jcgarcia     r     06/07/2011 11:11:26 cluster.q@
 652229 0.00000 cct_ref-al cnunez       qw    06/08/2011 13:00:18
 652230 0.00000 cct_ref-al cnunez       qw    06/08/2011 13:00:18
 652231 0.00000 cct_ref-al cnunez       qw    06/08/2011 13:00:18
 652232 0.00000 cct_ref-al cn
 652233 0.00000 cct_ref-al cn
 652234 0.00000 cct_ref-al cn
 652235 0.00000 cct_ref-al cn
 652236 0.00000 cct_ref-al cn
 652237 0.00000 cct_ref-al cn
 652238 0.00000 cct_ref-al cn
 652239 0.00000 cct_ref-al cn
 652240 0.00000 cct_ref-al cn
cnunez@clus33:~>
```

```
----------------------------------------------------------
Enqueued jobs status
----------------------------------------------------------

Job 652236  >>  finished normally          [cct_ref-aloha-DES-2.ef]
Job 652237  >>  finished normally          [cct_ref-aloha-DES-9.ef]
Job 652234  >>  finished normally          [cct_ref-aloha-DES-7.ef]
Job 652235  >>  finished normally          [cct_ref-aloha-DES-3.ef]
Job 652238  >>  finished normally          [cct_ref-aloha-DES-11.ef]
Job 652229  >>  finished normally          [cct_ref-aloha-DES-4.ef]
Job 652239  >>  finished normally          [cct_ref-aloha-DES-8.ef]
Job 652240  >>  finished normally          [cct_ref-aloha-DES-5.ef]
Job 652231  >>  finished normally          [cct_ref-aloha-DES-10.ef]
Job 652230  >>  finished normally          [cct_ref-aloha-DES-6.ef]
Job 652233  >>  finished normally          [cct_ref-aloha-DES-1.ef]
Job 652232  >>  finished normally          [cct_ref-aloha-DES-12.ef]

----------------------------------------------------------
All enqueued jobs finished
----------------------------------------------------------
```

Fig. 14. Command line vs. Cluster GUI Monitoring .

ClusterGUI Experience

ClusterGUI tool has also been successfully tested with other models projects implemented with OPNET. One of them is the Distributed Routing Balancing, DRB [7] , which is a method to uniformly balance communication traffic over the interconnection network. Other model executed with ClusterGUI is FT-DRB [8], a novel fault-tolerant routing method provided with a new deadlock avoidance technique designed to solve an unbounded number of faults appearing at random during system operation. MD-DRB [9], an extension to DRB was used with ClusterGUI as well. MD-DRB provides switches with capabilities such as escape paths and fast acknowledges generation mechanisms that lead to faster response time, when network is becoming congested. PR-DRB [10] was also used with ClusterGUI. PR-DRB uses speculative routing based on application repetitiveness. Also, it monitors messages latencies on routers and logs solutions to congestion, to quickly respond in future similar situations.

Conclusion

In this paper we presented ClusterGUI, an interface between OPNET simulations and a Distributed Resource Management (DRM) system. ClusterGUI is a tool capable of execute simulations in HPC like environments, where resources are generally shared among many users and jobs. ClusterGUI allows the execution of parametric simulations using all available resources, which are managed by a management entity.

Acknowledgments

This research has been supported by the MEC-MICINN Spain under contract TIN2007-64974. Furthermore, we thank OPNET Technologies, Inc. for providing us the OPNET Modeler licenses to perform the experimental evaluation of this work.

References

[1] R. Jain, "Book review: The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation, and modeling by raj jain (john wiley & sons 1991)," SIGMETRICS Perform. Eval. Rev., vol. 19, pp. 5–11, September 1991, reviewer-Al-Jaar, Robert Y.

[2] T. OPNET, "Opnet modeler accelerating network r&d," http://www.opnet.com, June 2008,OPNET.

[3] O. Corporation, "Oracle grid engine," http://www.oracle.com/technetwork/oem/grid-engine-66852.html.

[4] O. G. Forum, "Distributed resource management aplication api - drmaa," http://drmaa.org/, 2011.

[5]O. Corporation, "Java SE overview,"www.oracle.com/technetwork/java/javase/overview/index.html, April 2011.

[6] ——, "Java SWING. Creating a gui with jfc/java." download. oracle.com/javase/tutorial/uiswing/, April 2011.

[7] D. Franco et al., "A new method to make communication latency uniform: distributed routing balancing," in ICS '99: Procs of the 13th int. conf. on Supercomputing. USA: ACM, 1999, pp. 210–219.

[8] G. Zarza, D. Lugones, D. Franco, and E. Luque, "Fault-tolerant routing for multiple permanent and non-permanent faults in hpc systems," in International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'10), Las Vegas, NV, USA, July 2010, pp. 144–150.

[9] D. Lugones et al., "Dynamic and distributed multipath routing policy for high-speed cluster networks," in CCGRID '09: Procs of the 2009 9th IEEE/ACM Int. Symp. on Cluster Computing and the Grid, USA, 2009, pp. 396–403.

[10] C. Núñez, D. Franco, "Predictive and distributed routing balancing. PR-DRB." Master's thesis, Universitat Autònoma de Barcelona, July 2010.