UNDER REVIEW

RESCIENCE C

Replication / Computer Science

[Re] Velho and Legrand (2009) - Accuracy Study and Improvement of Network Simulation in the SimGrid Framework

Arnaud Legrand[1,2,3,4,5, ID] **and Pedro Velho**[6]

[1]Univ. Grenoble Alpes, Grenoble, France – [2]CNRS, Délégation Alpes, Grenoble, France – [3]INRIA Rhone Alpes, Grenoble, France – [4]Grenoble INP (Institute of Engineering Univ. Grenoble Alpes), Grenoble, France – [5]Laboratoire d'Informatique de Grenoble, CNRS UMR5217, 38000 Grenoble, France – [6]Ryax Technologies, Lyon, France

Edited by
(Editor)

Received
01 November 2018

Published
–

DOI
–

This paper reports the successful reproduction of the results in a article [1] entitled *Accuracy Study and Improvement of Network Simulation in the SimGrid Framework*, which has been published at the SimuTools 2009 conference. In this article, we detail several pitfalls we stumbled upon during this process and report the actions we took to improve the reproducibility of this work.
The first action we took is related to the visibility and availabiliy of this article. Open access was not mandated by the funders of this research at the time of publication and the only the bibliography entry was available on HAL. The preprint was hosted on our webpage and visible only through the SimGrid publication webpage.

Action #1: The PDF version of the original article has thus been uploaded on HAL when engaging in the replication process.

1 Historical Context

This article compares the accuracy of two methods for predicting how competing TCP network flows interfere with each others. It is the first article Arnaud Legrand wrote with his first PhD student Pedro Velho. It was already a reproduction of the work [2] of close colleagues, Henri Casanova and Kayo Fujiwara, and we already had faced difficulties in doing so at that time. Actually, we could never obtain the exact same numbers as them despite their care and ours. This failure motivated us to improve our methodology, and in particular switching to R, but it was 10 years ago. It was thus a good test of time!

1.1 Scientific context

SimGrid is a simulation toolking allowing to evaluate the performance of large scale distributed computing systems such as data grids, desktop grids, clusters or peer-to-peer systems. In this field, it is common to resort to simulation which enable to (in theory) reproducible results and allow to explore many application and platform scenarios, including platforms which do not exist yet. Unfortunately, as noted in [3], in this field most people build their own "ad-hoc" simulators which are rarely validated (, which makes final results quite questionable) nor made available (, which hinders both reproducibility of results and comparison of articles with one anothers). SimGrid is an attempt to provide a high quality simulation toolkit, which would be stable and perenial from a software point of view and whose models would be as validated as possible against reality and other simulators.

Correspondence should be addressed to Arnaud Legrand (arnaud.legrand@imag.fr)
The authors have declared that no competing interests exist.
Code is available at https://github.com/alegrand/reproducibility-challenge..

In this context, network simulation is certainly the most critical part and packet-level simulation are thus often considered as particularly realistic and faithful since they try to account for every detail of the network protocols. Unfortunately, such *microscopic* approach is undoubtedly interesting when studying peculiarities of network protocols, it leads to prohibitively long simulation time when studying large-scale distributed systems. An alternative is thus to simulate networks by relying on higer level, *macroscopic* models, thus enabling much faster simulation at the potential cost of an accuracy loss. SimGrid, uses a flow-level approach that approximates the behavior of TCP networks, including TCP's bandwidth sharing properties. A preliminary study of the accuracy loss by comparing it to popular packet-level simulators has been proposed in [2] and in which regimes in which SimGrid's accuracy was comparable to that of these packet-level simulators were identified. The article we reproduce here [1] was a reproduction these experiments and provided a deeper analysis that enabled to greatly improve SimGrid's range of validity.
The network is modeled as a graph where nodes represent hosts while edges represent network links. In SimGrid's flow-level modeling, the time needed to transfer a message of size S between hosts i and j is given by:

$$T_{i,j}(S) = L_{i,j} + S/B_{i,j}, \tag{1}$$

where $L_{i,j}$ (resp. $B_{i,j}$) is the end-to-end network latency (resp. bandwidth) on the route connecting i and j. Although determining $L_{i,j}$ may be straightforward, estimating the bandwidth $B_{i,j}$ is more difficult as it depends on interactions with every other flow. This is generally done by assuming that the flow has reached *steady-state*, in which case the simulation amounts to solving a bandwidth sharing problem, i.e., determining how much bandwidth is allocated to each flow.
More formally, consider a connected network that consists of a set of links $\mathcal{L}$, in which each link l has capacity B_l. Consider a set of flows $\mathcal{F}$, where each flow is a communication between two network vertices along a given path. Determine a "realistic" bandwidth allocation ρ_f for flow f, so that:

$$\forall l \in \mathcal{L}, \sum_{f \text{ going through } l} \rho_f \leq B_l \ . \tag{2}$$

In SimGrid, the "realistic" bandwidth sharing model [4] used is Max-min fairness [5], which is reached by recursively maximizing

$$\min_{f \in \mathcal{F}} w_f \rho_f \quad \text{under constraints in Eq. (2)}, \tag{3}$$

where w_f is generally chosen as the round-trip time of flow f. This objective corresponds to what one would naively expect from a network, i.e. be "as fair as possible" so that the least favored flows receive as much bandwidth as possible while accounting through weights w_f for the well-known RTT-unfairness of TCP [6].
Given the computed bandwidth allocation (which defines all data transfer rates), and the size of the data to be transmitted by each flow, one can determine which flow will complete first. Upon completion of a flow, or upon arrival of a new flow, the bandwidth allocation can be reevaluated. Compared to a packet-level simulation, this approach allows to quickly step-forward in time when large data transfers are involved. However, since steady-state is assumed, it ignores many transient aspects such as throughput oscillations, and slow start. This work was later extended to compare with other bandwidth sharing models [7] and has been the core of the PhD thesis of Pedro Velho [8].
In the article we reproduce, the accuracy of the *flow-level* simulations of SimGrid are compared to the *packet-level* simulations of GTNetS [9] the Georgia Tech Network Simulator. This was done through three series of simulations

1. One-link: The first set of experiments is for a single TCP flow going through a single link with varying physical **latency** and **bandwidth**, and message **size**. The

main goal of this scenario is to study the size for which transient effects such as slow start are negligible.

2. A Dumbell Topology: The second set of experiments is for two TCP flows A and B on a dumbbell topology with varying **bandwidth** of the inner link and **latency** of the end-link used by flow B. The main goal of this scenario is to study the ability of accounting for RTT-unfairness.

3. Random Topology: 4 sets of 10 random topologies generated with two topology generators were used. The sets comprised either small (50 nodes) or large (200 nodes) and either relatively homogeneous or heterogeneous platforms. 200 flows were generated between random pairs of end-points in the topology, which all start simultaneously and communicate 100MB of data. The main goal of this scenario is to evaluate the overall accuracy of SimGrid and possibly to detect corner-case situations for which the SimGrid model was particularly wrong.

Due to the long simulation time, we only reproduce in this article the first series of simulation but we checked that we could easily run at least one simulation of the two other series.

1.2 Computational context

SimGrid is mostly written in `C` while GTNetS is mostly written in `C++` and both are open source simulators. Although SimGrid is designed to be as stand alone as possible GTNetS relies on third party libraries. **The first challenge would thus be to reproduce a software environment allowing to recompile and rerun both libraries**.
To ease the comparison of both simulators, SimGrid had been modified to run GTNetS internally, which allowed to easily switch between the microscopic (GTNetS) model and the macroscopic (Max-Min) model from the command line, while using the exact same platform description and communication scenario. This integration required modifying both SimGrid and GTNetS and was done through a set of patches before being integrated in the main branch of SimGrid. **The second challenge would thus be to manage to correctly modify and recompile a simulator using both libraries.**
Although these details were not given in the articles, the general workflow of the simulations for all three scenarios was as follow:

- A simple `C` code called `gtnets.c` was linked against SimGrid and GTNetS;
- A `perl` script called `sweep-parse.pl` (when called with the `sweep` argument) would generate platform and flow/deployment `XML` input files and run all simulations by passing the previous `XML` input file to the `gtnets` binary with a different command line argument to switch between the GTNetS model and the Max-Min model. The simulation would produce a text output.
- The same `perl` script (when called with the `parse` argument) would then parse all the text logs and produce a `csv` data file.
- The data file would then be analyzed with an `R` script and since our mastery of `R` was quite low at that time, we still relied on gnuplot to generate figures.

The third challenge would thus be to manage to run all this workflow, provided the right instructions could be found.
Note that although the first two series of experiments did not have much external dependencies, the third one relied on many random network topologies generated by BRITE [10], which is a dicontinued `Java` software, using the Waxman model [**Waxman**]. The description of the parameters used to generate the topologies were shallow and there was no information regarding seeds so our hope to rerun this software to regenerate the same topologies was quite low. However, theses intermediate files may have

been stored and made available. **The fourth challenge would thus be to recover the network topology and data used in the third series of experiments**.

2 Rebuilding the code and its environment

2.1 Original source code and retrieval of the software

Instructions — Although the development of SimGrid is still very active, GTNetS' development appears to be discontinued as the last version of GTNetS dates back October 2008. Finding both source code is relatively easy however, the main difficulty was to find the instructions and to know which version to use. SimGrid has successively moved from the Inria gforge to the Inria gitlab, GitHub, and more recently Framagit. Although the whole software history has been correctly moved in the process, we realized some information have not been transfered and even sometimes lost:

- Although we could have used a development version of SimGrid from late 2009, we thought it would be simpler to reproduce this work using a stable release (e.g., the version 3.3, which dates from April 2009). Unfortunately, the releases of SimGrid on GitHub only start from May 2010. Indeed, although the SimGrid git history starts from 2004 (, when migrating from CVS to SVN), when the development team decided to migrate from subversion to git (in 2010), the SVN tags have not been transfered. Fortunately, the old releases of SimGrid are still available on the Inria gforge.

  > *Action #2:* We have thus now uploaded the original release of SimGrid version 3.3 on Github.

- The LaTeX source of the article is stored in the private Inria Gforge simgrid-publis project, in an `svn` under the `PUBLISHED/09_validation_simutools` directory.

  > *Action #3:* We have now made the LaTeX source of the article available in the github repository attached to this Rescience submission.

- We know that we made our instructions on how conduct these experiments available somewhere but no link was given in the original article and we could not really remember where it was as there was no standard way of doing so back then. We though they were given on the former `contrib/` section of http://simgrid.gforge.inria.fr/ (, which was hard to maintain and was thus abandonned) or on http://simgrid-publis.gforge.inria.fr/ (, which finally only hosts data on two articles from 2011). However after inspecting the Internet Archive, we could not find it.

- Arnaud Legrand therefore tried them on his laptop but although he could find many related files (including the topology generators) he failed finding the right data and doing so, he realized many the data of some of his previously published articles were dangling links and had not been correctly transfered when migrating from a laptop to an other! The instructions could probably have been recovered on backup hard drives but he had the chance to meet Pedro Velho and to ask him whether his own backups were in better shape, which was fortunately the case. Pedro Velho could find all the required data (a 61MB zip archive) and shared it through dropbox. This data may be still available somewhere on the Internet but as we had recovered it, we have not put additional efforts.

  > *Action #4:* We have now made the instructions and data used in the original article available in the GitHub repository attached to this Rescience submission.

simutools09/instructions/README

```
Author        : Pedro Velho
last modified : 03/11/2008

#################################
 Disclaimer
#################################

This text as well as the data and results provided here are under
GPL copyright. To consult the GPL terms and usage condition see in the
top directory:

GPL.txt

All programs use the gtnets.c simulation program. A source
code copy is located in this directory.

CAUTION:
This script relies on parsing the output, so every modification (even
slight ones) on gtnets.c output may cause the parsing feature to
unpredicted behavior. If you are not sure about your gtnets.c file
please use the one distributed here.


#################################
 Short History
#################################

This directory hold experiment comparing the SimGrid framework
network simulation engine with GTNets. GTNets is a packet level
network simulator and we believe it can provide realistic
transmission time prediction due to its characteristic of
simulating through discrete events the entire TCP protocol stack.
This work is the normal continuation of the work presented
by Kayo Fujiwara and Henry Casanova in [1] and was submitted to
the SimuTools09 conference which is still to be evaluated and
accepted/reject.

#################################
 Directory Structure
#################################

Three category of experiments were performed. Each one was tackled
separately and are organize in distinct directories as follows:

* ./01-onelink  - Verify message size communication time correlation

* ./02-dumbbell - Bandwidth sharing experiments

* ./03-random   - Complex platforms to assure model improvements


#################################
 Global System Requirements
#################################

Experiments and analyze scripts are provided to reproduce the
graphs presented in [FIXME(Rapport de recherche ou reference
simutools]. To run experiments some minimum system requirements are
needed:

* GTNets patched simgrid version, we kindly provide GTNets with
  patches in the simgrid contrib svn repository [FIXME]

* SimGrid, configured and compiled with GTNets support [FIXME]

For plotting graphs and explore the data:

* R - the gnu version of S [FIXME]
* Gnuplot [FIXME]

#################################
 Refernces
#################################

[FIXME] The R (GNU S) language website. FIXME URL

[FIXME] Plotting scientific data with Gnuplot. FIXME URL

[FIXME] SimGrid website. FIXME URL

[FIXME] Kayo Fujiwara and Henry Casanova FIXME
```


Figure 1. dfd

This archive comprises 3 sub-archives corresponding to each of the 3 series of simulations mentioned earlier (`01-onelink.tgz`, `02-dumbbell.tgz`, `03-random.tgz`) as well as a GTNetS version (`GTNetS-Oct-10-08.tar.gz`) and the master simulation file (`gtnets.c`) which should be compiled against SimGrid and GTNetS. The README that can be found in each subarchives describes in details how to rerun the experiments and corresponds to the process described in Section 1.2. Unfortunately, the master README (see Figure 1 provides information about dates and the contents of the archive but most information related to software versions are broken (it was a working version, which we intended to consolidate when the article would be accepted). Furthermore, after a thorough inspection of the GTNetS archive, we realized it did not seem to have been patched.

- Arnaud Legrand therefore started searching again for gtnets versions that would be on his laptop and finally found it, along with all the patches and compiling instructions which are crucial to correctly build such prototype software (see Figure 2. These information were actually public but had become completely hidden in the (now unmaintained and long forgotten) contrib section of the SVN (while git is now the default version manager) of the Inria Gforge SimGrid project.

Action #5: **We have now ensured that the GTNetS version and the patches we used are archived on Software Heritage.**[a]

[a]The save request was done on 4/30/2020, 6:50:02 PM but it is still pending.

In the end, we have thus managed to recover three important archives, whose versions should be the one run to produce the results of the original Simutools09 article:

1. The stable release `v3.3` of SimGrid (from April 2009) from the public Inria Gforge. Although experiments were probably run in late 2008, the previous stable SimGrid release is from 2007 and `v3.3` incorporates everything that was needed.
2. A snapshot of GTNetS from January 2008 along with the patches to apply from the public Inria Gforge but which was not visible anymore.
3. The simulation instructions and data from one of the author's hard drive.

No information regarding the software environment is available except that it was run on a Debian in the late 2008.

2.2 Rebuilding the software environment

SimGrid is mostly a C library whose software dependencies had at that time been kept to the bare minimum (C and C++ compiler). Furthermore, we are developers of the SimGrid library so building it was rather straightforward. However, after trying to compile GTNetS, we realized it depends on the Qt3 GUI Library whereas the version which is now commonly found is Qt5! Therefore, we decided to recreate a minimal software environment as close as possible.
The codename for a stable Debian distribution at that time was *Lenny*. Debian provides two particularly interesting tools to reproduce "old" environments:

1. The snapshot archive is a wayback machine that allows access to old packages based on dates and version numbers. It consists of all past and current packages the Debian archive provides.
2. The Debuerreotype is a reproducible, snapshot-based Debian rootfs builder. It allows to prepare from old debian images from the snapshot archive, which is particularly useful to build Docker images containing old software environments.

simutools09/README.patching_GTNetS

```
==================================
GTNetS/Simgrid patch README
author: Pedro Velho
==================================

Note About this Patch
---------------------
This patch is intended to work only with GTNetS downloaded from the GTNetS website link:
http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/software/gtnets-current.zip
The last time this patch was downloaded was June 12 2008, Seems they don't have much control
about new GTNetS and some other flaws it is difficult to precise a verison number.

Getting GTNetS
--------------
Two ways of getting GTNetS, one from the gtnets website and svn
simgrid contrib projec tree (RECOMMENDED).
$ wget http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/software/gtnets-current.zip
or
$ svn checkout svn://scm.gforge.inria.fr/svn/simgrid/contrib/trunk/GTNetS/

Applying the PATCH
------------------
$ unzip gtnets-current.zip
$ tar zxvf gtnets-current-patch.tgz
$ cat *.patch | patch -p1

Compilling GTNetS
-----------------
Enter directory
$ cd gtnets-current
GTNetS is not a very active project for the moment and the portabillity is really limitated.
For the moment we tried out this patch only in linux platforms using:

gcc (GCC) 4.1.3 20070629 (prerelease) (Debian 4.1.2-13) Linux 2.6.21-2-686

Create a Makefile.linux symbolic link
$ ln -sf Makefile.linux Makefile
Create dependecies list
$ make depend
To compile debug version
$ make debug
To compile optimized version
$ make opt

=== WARNING ===
A lot of warnings are expected but the application should compile just fine. If the makefile
insists in compiling some QT libraries please try a make clean before asking for help.

Installing GTNetS
-----------------
Commands make debug and opt generates respectivelly libgtsim-opt.so or libgtsim-debug.so.
You will need to link ONLY ONE of these libraries using the simbolic link name libgtnets.so,
for instance to libgtsim-debug.so:
# ln -sf libgtsim-debug.so /<userhome>/usr/lib/libgtnets.so
'libsimgrid.so' is the name simgrid is configured to search when running ./configure script.
Now just put the library somewhere you know ldd is searching  (tip:
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/<userhome>/usr/lib/libgtnets.so && sudo ldconfig)

The gtnets source headers are necessary. So you need to copy all headers to a place where
your compiler can find them, such as:
$ mkdir /<userhome>/usr/include/gtnets
$ cp -fr SRC/*.h /<userhome>/usr/include/gtnets

Compilling SimGrid with GTNetS
------------------------------
Just add the following option when running configure

--with-gtnets=/<userhome>/usr

Bug reports, comments, suggestions: pedro.velho{\at}imag.fr

AMD64 bit patch
---------------
Some users experienced some problems during compilation on AMD64 bit architecture.
We compiled succesfully the gtnets-current package in an:

model name      : AMD Opteron(tm) Processor 248
stepping        : 8
cpu MHz         : 2193.160
cache size      : 1024 KB
clflush size    : 64
cache_alignment : 64
Using gcc (GCC) 4.2.3 (Debian 4.2.3-3)

We provide a simple patch to do this: AMD64-FATAL-Removed-DUL_SIZE_DIFF-Added-fPIC-compillin.patch
```


Figure 2. dfd

Pedro and myself regularly used `testing` so after investigating a bit on the snapshot archive which versions of the libraries and when they had been introduced, we decided to try to bootstrap a debian Lenny from the 1st of May 2009 with the following command:

```
debuerreotype-init --keyring=/usr/share/keyrings/debian-archive-removed-keys.gpg \
   rootfs testing 2009-05-01-T03:27:08Z
```

Building such an image involves installing (with `dpkg`) old packages in a subdirectory pretending you are root. The `keyring` argument passed to `debuerreotype-init` allows to indicate `dpkg` that it is safe to install these old packages even if they have been signed by package maintainers which are currently not active anymore. Unfortunately, although this approach worked like a charm for more recent target dates (e.g., `2015-06-04-T10:47:50Z`), it miserably fails with a "Segmentation fault" when installing `base-passwd`:

```
W: Failure trying to run: chroot "/home/alegrand/Work/Documents/Articles/2020/
        reproducibility_challenge/simgrid3.3_gtnets/rootfs" dpkg --force-depends
             --install /var/cache/apt/archives/base-passwd_3.5.21_amd64.deb
W: See /home/alegrand/Work/Documents/Articles/2020/reproducibility_challenge/
         simgrid3.3_gtnets/rootfs/debootstrap/debootstrap.log for details

error: 'debootstrap' failed!
```

We then decided to cry for help and asked two Debian guru friends, Vincent Danjean and Samuel Thibault. Samuel indicated me that he would investigate this by simply using

```
debootstrap wheezy myroot http://archive.debian.org/debian/
```

and that the error message was then slightly more visible

```
dpkg: warning: parsing file '/var/lib/dpkg/status' near line 5 package 'dpkg':
 missing description

Package: dpkg
Status: install ok installed
Maintainer: unknown
Version: 1.16.18
```

When bootstraping such an image, we try to use old debian packages with a recent `dpkg` (the one running on our machine) so it is not surprising that it may break. After all, the internal format of Debian packages could have evolved and may not be supported anymore with recent versions of `dpkg`. Likewise, it is somehow a matter of luck that an old binary still works with a recent kernel... Indeed, when using docker or similar container-based approach, we only divert syscalls so if the ABI of the Linux kernel had changed in the meantime, binary codes would simply fail to run. Fortunately, such changes are quite rare and the Linux/Debian community is making incredible efforts to provide super stable backward compatible software so what could be the reason behind this failure?
Surprisingly Vincent Danjean reported me that the command worked like a charm for him, which means some local configuration of my or from his machine could change this behavior. We could actually track back the problem to an ABI modification of the kernel. As explained for example on the Einstein@Home forum, *"On latest Linux distros,* `vsyscall` *is defaulted to none for security reasons. However, this breaks some very old binaries, including some binaries from this project that are statically-linked against ancient versions of glibc"*. Vincent had activated this a long time ago to run some old proprietary code. Booting the machine while adding `vsyscall=emulate` to the kernel command line allows `debuerreotype` to build the desired rootfs.
Since this is a bit far-fetched, we decided to check whether ready-to-use Docker images were available on the Docker Hub, which is the case. After playing a bit interactively in this Docker image trying to install everything we needed to build GTNetS and SimGrid, and following the patching and compiling instructions, we ended up with the Dockerfile presented in Figure 3. The image can be simply build with the following command:

simutools09/simgrid3.3_gtnets/Dockerfile

```
FROM lpenz/debian-lenny-i386

LABEL maintainer="Arnaud Legrand <arnaud.legrand@imag.fr>"

# Software dependencies
RUN apt-get update \
        && apt-get install -y --force-yes gcc g++ make wget unzip subversion patch \
           less libqt3-mt libqt3-headers libqt3-mt-dev qt3-dev-tools

# Code dowloading and assembly dependencies
RUN apt-get update \
         && apt-get install -y --force-yes wget unzip subversion patch less

# Downloading GTNetS
RUN cd /root; svn checkout svn://scm.gforge.inria.fr/svn/simgrid/contrib/trunk/GTNetS/
# Downloading SimGrid
RUN cd /root; wget https://gforge.inria.fr/frs/download.php/file/21430/simgrid-3.3.tar.gz

# Building GTNetS
RUN cd /root/GTNetS/; unzip gtnets-current.zip ; tar zxvf gtnets-current-patch.tgz
RUN cd /root/GTNetS/gtnets-current; cat ../00*.patch | patch -p1
RUN cd /root/GTNetS/gtnets-current; ln -sf Makefile.linux Makefile && make depend && make opt

# Installing GTNetS
RUN cd /root/GTNetS/gtnets-current/ && \
    mkdir -p /root/usr/lib/ &&         \
    ln -sf `pwd`/libgtsim-opt.so /root/usr/lib/libgtnets.so && \
    ln -sf `pwd`/libgtsim-opt.so /usr/lib/libgtnets.so &&      \
    mkdir -p /root/usr/include/ &&    \
    cp -fr SRC/*.h /root/usr/include/

# Building SimGrid
RUN cd /root/ && tar zxf simgrid-3.3.tar.gz
RUN cd /root/simgrid-3.3/ && \
    ./configure --with-gtnets=/root/usr/ && \
    export LD_LIBRARY_PATH=/root/usr/lib/libgtnets.so && \
    ldconfig && \
    make

RUN apt-get clean
```

Figure 3. dfd

```
docker build -t alegrand/simgrid3_3_gtnets simgrid3.3_gtnets
```

Action #6: We have now an automated way to build a minimalistic environment comprising the simulation code used in the original article. This Dockerfile recipe has been made available in the GitHub repository attached to this Rescience submission.

3 Execution and reproduction of results

4 Conclusion

- Using the right tools to automate (R, perl,
- Automated workflow/notebook to document the process
- Woul someone else have managed reproduce the work ?
- Cleaning up is rarely done after publishing, hence the need to do it on the fly.
- Perenial URLs
- Software environment

References

1. P. Velho and A. Legrand. "Accuracy Study and Improvement of Network Simulation in the SimGrid Framework." In: **SIMUTools'09, 2nd International Conference on Simulation Tools and Techniques**. Rome, Italy, Mar. 2009.
2. K. Fujiwara and H. Casanova. "Speed and Accuracy of Network Simulation in the SimGrid Framework." In: **Intl. Conf. on Performance Evaluation Methodologies and Tools**. Nantes, France, 2007, 12:1–12:10.
3. S. Naicken, A. Basu, B. Livingston, and S. Rodhetbhai. "Towards Yet Another Peer-to-Peer Simulator." In: **Proc. Fourth International Working Conference Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETs '06)**. Sept. 2006.
4. H. Casanova and L. Marchal. **A Network Model for Simulation of Grid Application**. Tech. rep. 2002-40. LIP, Oct. 2002.
5. L. Massoulié and J. Roberts. "Bandwidth Sharing: Objectives and Algorithms." In: **INFOCOM**. 1999, pp. 1395–1403.
6. G. Marfia, C. Palazzi, G. Pau, M. Gerla, M. Sanadidi, and M. Roccetti. "TCP Libra: Exploring RTT-Fairness for TCP." In: **NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet**. Ed. by I. F. Akyildiz, R. Sivakumar, E. Ekici, J. C. d. Oliveira, and J. McNair. Vol. 4479. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 1005–1013.
7. P. Velho, L. Schnorr, H. Casanova, and A. Legrand. "On the Validity of Flow-level TCP Network Models for Grid and Cloud Simulations." In: **ACM Transactions on Modeling and Computer Simulation** 23.4 (Oct. 2013).
8. P. A. Madeira de Campos Velho. "Accurate and Fast Simulations of Large-Scale Distributed Computing Systems." Theses. Université Grenoble Alpes, July 2011.
9. G. F. Riley. "The Georgia Tech Network Simulator." In: **ACM SIGCOMM workshop on Models, Methods and Tools for Reproducible Network Research**. Karlsruhe, Germany, 2003, pp. 5–12.
10. A. Medina, A. Lakhina, I. Matta, and J. Byers. **BRITE: Universal Topology Generation from a User's Perspective**. Available at https://www.cs.bu.edu/brite/publications/usermanual.pdf. Apr. 2001.