Abstract

We present the LabWiki, an executable paper platform primarily designed but not limited to networking experiment-based research. The LabWiki leverages the current state of the art tools for the orchestration of experiments in the networking community and propose a new approach to execute and reproduce experiments. We demonstrate the usability of the LabWiki through an example at the boundary between network and high performance computing researches.

Keywords: Network, Scientific Method, Experiment based, OMF

1. Introduction and Motivation

Many publications in the field of computer networking increasingly contain experimentally-based evaluations of the proposed systems and algorithms. However, an informal study we conducted on last year’s accepted papers for one of the leading computer networking conferences indicated problems with 70% of the accepted papers related to proper description of methodology, experiments, and analysis [1]. While some authors now provide access to the collected measurements, shortcomings in capturing context and provenance limit their scientific value for evaluating the published results themselves as well as for further research.

We have developed a comprehensive set of tools for managing large scale experimental facilities, as well as supporting researchers in conducting experiments on these facilities. The resulting framework, called OMF [2, 3], is deployed in over 20, primarily wireless and mobile computing focused testbeds around the world and hundreds of papers have been published containing experiments conducted with our tools.

OMF is heavily focused on supporting repeatable experiments through OEDL[2], a domain-specific language fully capturing the experiment setup, its orchestration, and all relevant context; as well as a fully integrated instrumentation and measurement framework which not only efficiently collects all the artefacts of an experiment, but maintains full provenance by linking back to the experiment context.

While OMF supports a very systematic approach to experimental validation, we have seen very few examples where this follows through into the resulting publications. We have therefore recently embarked on the design and implementation of a Portal to support an entire investigation based on the hypothetico-deductive model[4] of the scientific method as shown in Figure 1. In this Figure, we present the different components of the portal and their equivalent in the scientific methods. In this Figure, we noted $H_a$ and $H_0$ the hypotheses and null-hypotheses derived from the models.
The current version allows a researcher to (re)run, track, and record experiments on managed testbeds as well as manage all resulting artefacts, such as scripts, software versions, and measurement databases. We adopted the ticketing functionality common to most software development Portals to fully capture each experiment in a single page with its own unique public URL. A button on the page allows the instantiation of a new experiment with the same parameters, making it very easy to re-run an experiment. The Portal is independent of any specific testbed installation and different third parties may host Portal instances interfacing with different types of testbeds. Its current implementation is compatible with any testbed running OMF (such as Orbit Lab, NITLab, NICTA Norbit [5, 6, 2]).

In response to this challenge we propose LabWiki which is essentially an extension of our current Portal with a laboratory notebook like functionality based on the Portal’s wiki capability. This allows the researcher to describe the problem studied, the models proposed and the hypothesis and anti-hypothesis motivating the design of the various experiments. Each of the artefacts of an experiment and the experiments themselves are identified by public URLs which can be linked easily to from any LabWiki page. We further propose to use the R [7] language to analyse any measurements collected on the Portal. The R scripts are similar to the OEDL scripts mentioned previously and we will use the existing Portal functionality to archive, run and also re-run analytic tasks the same way we do experiments. The results, such as graphs or tables, can be embedded directly in a LabWiki page thanks to an extended wiki syntax.

With all this in place, any LabWiki page can become an executable paper. In fact, other LabWiki pages can be used to capture the resulting review process including the verifying experiments and additional analysis of the measurements.

This remainder of this paper is as follows. Section 2 presents the general architecture of the Portal and its different features and the components that support researchers in addressing this experimentation issues. Then Section 4 goes though a study case where we investigate how to use the LabWiki to analyse the impact of the communication between entities in a P2P computational platform. Section 5 go through the challenge criteria and how our proposed solution answers them. Finally, Section 6 concludes and gives further research directions.

2. Background

In this section, we present a brief overview of both the testbed management framework OMF [3] and its companion measurement library OML [8]. In the context of the LabWiki, the measurement library serves two purposes; storage of user-defined measurements, as explained in the remainder of the section and more importantly the capture of environment information. This environment data is also made available in the LabWiki and aims at maximising the reproducibility of experiments.
OMF - Experimental facilities (or testbeds) are instrumental for the evaluation of new network technologies. In many cases, these testbeds are solely built and used for a specific research project, and are often not maintained, reused, or shared. This obviously limits the independent verification of experimental results by the community which is a cornerstone of the scientific method. Beside access to a facility we also need an unambiguous way to describe an experiment and all its required resources to enable others to repeat it. To raise to this this challenge, we developed OMF [2, 3], a suite of management, control, and measurement services and tools for networking testbeds. From an operator perspective, OMF provides several services to manage, allocate and configure heterogeneous resources within a testbed. From an experimenter’s perspective, it provides a high level domain-specific language to systematically describe an experiment (i.e. its used resources, its required measurements, and its task to perform), and a set of software tools to automatically deploy and orchestrate this experiment on a given testbed.

Figure 2: Overview of OMF architecture from the user’s perspective (source: [2])

Figure 2 shows a simple overview of OMF’s architecture from an experimenter’s point of view. As described on this figure, the input to the OMF system is an Experiment Description, which is produced by the researcher (i.e. the user). OMF will then perform all the necessary operations to deploy, configure, and execute the corresponding experiment. While the experiment is running, various measurements are automatically collected through the use of the OML measurement library. OMF and OML have been deployed on several heterogeneous testbeds, and have been used by many researchers worldwide [2].

OML - OML [8] is an instrumentation & measurement framework, which was first developed as a component of OMF, but is now a stand-alone software which collects and stores any type of measurements from any type of application. OML has three components that allow the user to automatically generate and store measurement streams. First, a developer defines Measurement Points (MPs) within their applications or services. At run-time the experimenter can request these MPs to generate Measurement Streams (MSs) which can be further processed (e.g. filtered or combined), cached, and ultimately streamed towards repositories to be stored in databases for further analysis.

OML has been integrated in many applications, such as traffic generators, passive network measurements, GPS coordinate loggers, and pressure/temperature sensor loggers [9]. We developed a measurement analysis component as part of the proposed LabWiki, which automatically generates simple graphs from the measurement database of a given experiment, and allow the import of these measurements into a wiki-based statistical analysis tool.
3. LabWiki Overview

Based on the existing reservation portal of the NICTA testbed [1], we developed LabWiki which allows researchers to easily apply the hypothetico-deductive approach [4] to their investigations. In addition, LabWiki facilitates the collaboration between researchers and peer-verification of final results through the use of a fine-grained shareable wiki pages.

Figure 4 illustrates the workflow of a typical iteration of a research investigation using the proposed LabWiki. This workflow is following the approach described in Figure 1. It starts with the formulation of a model based on existing observations or data, and the subsequent derivation of hypotheses from it. LabWiki supports these initial steps by providing a wiki based space which allows researchers to capture these models and hypotheses, and to potentially share them with selected collaborators. The researchers would then describe some experiments aiming at confronting these hypotheses, using the OEDL-based experiment editing interface of LabWiki. These experiments will then be deployed and performed on OMF-enabled testbed, using both the scheduling mechanisms of the original portal on which LabWiki is built on and the experiment control features of OMF. At this stage, one of the key contribution of LabWiki is that in parallel to the experiment-defined measurements, it will also automatically initiate the collection of additional context relevant measurements on the load and state of the resources being used (e.g. cpu/mem usage of resource X, channel quality seen by resource Y, etc...). All these measurement collections are done using the OML software described earlier [8]. Once the experiments are finished, the researchers may use the provided graphical interface to analyse the resulting collected data. This interface is the other major contribution of LabWiki. It allows the researchers to edit or load R scripts [7] describing statistical computations to be performed on the collected data. LabWiki will run these scripts into a R interpreter which has access to the experiment data, and will present the resulting outputs (e.g. graphs, tables,...) to the researchers.

Finally when the researchers decide to publish their results, LabWiki allows them to selectively mark as public the relevant parts of their LabWiki investigation. These public parts are accessible to any peers or reviewers and would in effect embody the executable paper per se, as these third parties would be able to use LabWiki to reproduce the experiments in similar contexts and verify the published results.

4. Study Case: Effect of Delay on the resolution of the Obstacle Problem

In this section we will present an example of a real research investigation using the LabWiki portal to demonstrate the feasibility of the previously described workflow and the versatility of LabWiki. This example is based on a problem which is at the border of both networking and high performance computing research. The hypothetical research project investigates the effect of the communication delay on the performance of the resolution of an obstacle problem [10] deployed over a P2P computing architecture called P2PDC [11]. In this article, we are merely presenting the methodology of such a study and do not aim at demonstrating any breakthrough research. Therefore the comprehensive results and analyses related to this study will be presented in a separate paper.
4.1. Model and hypotheses

The discretisation of the obstacle problem leads to the formulation of the following fixed point problem that can be solved by distributed iterative algorithms:

\[
\begin{aligned}
\text{Find } u^* \in V \text{ such that } \\
u^* = F(u^*)
\end{aligned}
\]

where \( V \) is an Hilbert space and the mapping \( F : v \rightarrow F(v) \) is a fixed point mapping from \( V \) into \( V \).

Many equivalent formulations of this problem can be found in the literature such as the complementary problem, variational inequality and constrained optimisation problem; the reader is referred to [12, 13] and [14] for more details.

In order to solve the problem (1), we will use the P2PDC architecture [11]. This architecture will use the Richardson algorithm combined with several classical synchronous schemes of computation like Gauss-Seidel or asynchronous schemes of computation[12].

The P2PDC architecture offers a P2P computational platform that can be deployed over numerous networking technologies including the Internet Protocol [11]. Therefore, this architecture offers new communication schemes to perform high performance computing without being tied to a grid architecture over dedicated network.

In the context of the P2PDC deployment over the Internet, we cannot expect the same efficiency and on-going increase of the speedup as we could have in a dedicated grid. Indeed, one could envision that the communication component of the distributed computation will not be negligible and could result in a loss of efficiency when the ratio time to iterate to the time to communicate reach a certain threshold. Therefore, we hypothesised the following possible behaviours of the speedup in function of the delay between peers:

\[1\] For implementation details please refer to [11].
- Speedup follows $\log(N)$ where $N$ is the number of peers,
- Speedup reaches an optimum level,
- Speedup follows $K.N$ where $N$ is the number of peers.

Furthermore, the scheme may not be universal and might depend heavily upon the network impairments. In this context, we call network impairment the communication delay between peers and the ratio of messages that will be lost. In the remainder of this section, we will show step by step the configuration and the executions of experiments to find the correct model according to the network.

4.2. Experiment Description

In order to evaluate the different models, we first need to feed the portal with experiment descriptions using the OEDL scripting language. Listing 1 presents a simplified version of the OEDL script that will be executed on the testbed. In this script the first five lines represent the parameters the experimenter will be able to tune for every run of the experiment. As stated in the previous section, we are interested in the effect of the network impairments on the efficiency of the different classes of algorithms, therefore we limited the parameters to the delay, the packet loss rate, the number of peers and the class of algorithms.

```
# Properties to change during the different experiments
defProperty ('nbPeers', 2, "the number of peers involved")
defProperty ('classAlgo', 'asynchronous', "the type of algorithm to use")
defProperty ('delay', '10ms', "the delay between peers")
defProperty ('plr', 0.0, "the packet loss rate between peers")

# Coordinator
defGroup('submitterGroup', 'node0') do |node|
  node.addApplication ('P2PDCAppSubmitter') do |app|
    app.measure('mp_submitter', :samples => 1)
  end
end

# Computing Peers
defGroup('donorGroup', '[[node1... node#{nbPeers}]]') do |node|
  node.addApplication ('P2PDCAppDonor') do |app|
    app.measure('mp_worker_result', :samples => 1)
    app.measure('mp_worker_diff', :samples => 1)
  end
end

# Topology
defTopology do |topo|
  topo.addNode("myNode_1", prop.node1)
  topo.addNode("myNode_2", prop.node2)
  # We describe the characteristics of the links between node 1 and 2
  topo.addLink("myNode_2","myNode_1",:emulationTool => :netem,:asymmetric => true,
    :ruleID => 3, :delay => prop.delay,:bw => '1Mbits', :bwBuffer => 12000,
    :bwLimit => 15000,:loss => prop.plr)
  topo.addLink("myNode_1","myNode_2",:emulationTool => :netem,:asymmetric => true,
    :ruleID => 3,:delay => prop.delay,:bw => '1Mbits', :bwBuffer => 12000,
    :bwLimit => 15000,:loss => prop.plr)
  topo.saveGraphToFile()
end

onEvent(:ALL_UP_AND_INSTALLED) do |event|
  group('donorGroup').startApplications
  wait 5
  group('submitterGroup').startApplications
  # Wait for application execution
  wait 1200
  # Stop the experiment
  Experiment.done
end

Listing 1: Experiment Description Example
The remainder of the experiment script describes the different components of the experiment. First we select one of the peers as the coordinator of the computation. This peer ensures that the algorithm finishes correctly. Then we deploy the computational algorithm to the number of peers previously configured.

Next we configure the network according to the different parameters. Without loss of generality, we limit the topology description to a single pair of peers. In the actual study these impairments will be applied between every peer in a first time and then on selected peers. In the presented script, we make the management framework to configure the link between the node 1 and 2 with the configured delay and message loss rate ($plr$ in the listing). Then we apply the same impairments to the link between 2 and 1.

Once physical and upper level definitions are finished, the user describes the logical and temporal orchestration of the experiment, which is represented by the last part of the script.

The description of the experiments has been completed using the OEDL scripting language, the user is now able to schedule the different runs. In the context of the presented study, we planned to made the properties vary as described in the Table 1. These different ranges result in the configuration of more than 2000 experiments which would be very time-consuming without the LabWiki.

<table>
<thead>
<tr>
<th>Table 1: Ranges for the Experiment Runs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Range</strong></td>
</tr>
<tr>
<td>Delay intra-cluster [1, 100] ms</td>
</tr>
<tr>
<td>Delay inter-cluster [10, 250] ms</td>
</tr>
<tr>
<td>Packet Loss Rate intra-cluster [0.0, 0.25]</td>
</tr>
<tr>
<td>Packet Loss Rate inter-cluster [0.0, 0.25]</td>
</tr>
<tr>
<td>Class of Algorithm {Synchronous, Asynchronous, Hybrid}</td>
</tr>
<tr>
<td>Number of Peers {1, 2, 4, 8, 16, 24, 32, 40}</td>
</tr>
</tbody>
</table>

4.3. Runs and Observations

Using the LabWiki, the user can configure all their experiments using a web interface automatically generated based on the experiment description presented in Listing 1. The Figure 5 (a) presents the resulting configuration web page. As specified in the experiment description, the user can configure the network impairments, the resolution algorithms and the number of peers.

Once the experiment is configured, it is put in a queue as described by the portal workflow and will be automatically run according to either the user’s reservation or the testbed availability. The user can then see the experiment queue in the Experiments tab of the LabWiki as illustrated in Figure 5 (b). In order to guarantee privacy between projects, a user can see only the experiments configured for a particular project.

4.4. Analyses

The last step in any research project is the analysis of the measurements. Using the LabWiki, a user can either download the measurement databases generated by OML and automatically attached to the result page or directly load and perform analysis of these measurements using the wiki interface. Indeed, as illustrated in Figure 6, we have enhanced the basic wiki syntax in order to allow the user to load the aforementioned measurement results directly in the wiki and then use the R language to analyse them.

In the context of the presented study case, as it is not the purpose of this article to present new results in the field of distributed computing, we present in Figure 6 a hypothetical result of the speedup of one class of algorithm in the case of a low delay and no packet loss rate.

5. Challenge Criteria

This section addresses the criteria listed in the Challenge brief.
5.1. Executability

The Portal stores experiment scripts alongside with any parameters used, and also holds and processes the experiment results and captures all state. Equations, tables, graphs are generated as part of the experiment flow and may be
displayed as part of a page in the LabWiki. After the experiment development process is finished, a Portal user may choose to open up his or her LabWiki permissions to specific reviewers or the general public. The readers can then review the experiment description and the result set as well as re-run, clone or manipulate the experiments to validate or extend the result sets inside their own Portal account.

5.2. Short and long-term compatibility

OMF experiments are based on the OEDL Language [2]. Since its introduction in 2002, OEDL is constantly being extended, but retains backward compatibility given the already large user community. Since experiments described in OEDL typically do not include any platform or operating system specific commands, and since OMF is written in the platform-independent Ruby language, we can ensure interoperability of OEDL experiments across a large number of computer systems.

5.3. Validation

Since the reviewer can access the experiment description, the applications and the experiment results through the Portal, he or she can easily validate the experiments that have been run and verify the data as well as its analysis that was produced for the research paper.

5.4. Copyright / Licensing issues

Copyright and licensing are primarily non-technical issues. However, to enforce and implement such policies, the Portal offers a granular permission system. It allows, for example, the authors to limit the access to their experiment descriptions and result sets to reviewers only during the review phase, and later open up permissions to the general public or other user groups after the paper has been accepted or published. Being able to safely publish the data that accompanies a research paper with just a few mouse clicks encourages authors to do so. Limiting access to closed user groups can be facilitated to protect intellectual property. We have also recently been evaluating new tools based on ‘differential privacy’ techniques to allow more open analysis of sensitive data collected from experiments involving real users which often require ethics board approval.

5.5. Systems

If measurements were conducted on a large scale computer system, where computing resources are limited or only available to certain users, reviewers and readers still have the opportunity to download a copy of the experiment description from the Portal and run it on a different, more accessible testbed. By adjusting the experiment parameters (e.g. the number of participating nodes), similar results may be achieved even on smaller testbeds, which in some cases may be sufficient to validate the author’s claims.

One of OMF’s basic principles is repeatability of experiments. If testbed resources are not available to someone who would like to repeat the experiment, it’s rather a policy issue of the testbed operator than a technical challenge.

5.6. Security considerations

In a testbed running OMF, all code is executed on isolated filesystems or virtual machines. Should malicious code enter the experiment description and applications, its execution would be limited to the set of nodes in the experiment. By default, the filesystem image on the nodes is wiped by OMF in between experiment sessions of different users.

Code that the user may have entered in the LabWiki is executed in isolated environments (jails), preventing interference with the Portal and the data stored in it.

5.7. Feasibility

Thousands of experiments have been conducted using OMF on more than 20 testbed facilities since 2002, hundreds of published papers include results gained through OMF managed testbeds. In addition, an early version of the Portal has been used in an e-learning context at a university in 2010, where students ran 9000 experiments on the IREEL platform [15] which is layered on top of OML. A paper discussing the outcome of this project and the lessons learned has been submitted.

The Portal implementation will be completed in early 2011 and will be deployed on a number of sites that already run OMF testbeds. The Portal itself only needs a web server and security contexts with the associate OMF testbeds. We aim to give a live demonstration of a full executable paper development cycle using the Portal shortly.
6. Conclusion

We presented LabWiki, a web portal designed to support the full life-cycle of experiment-based investigations. It interfaces with OMF, a testbed management as well as experimenter support framework designed with a focus on experiment repeatability and sound data collection.

The current version of LabWiki already allows an authorised user to re-run experiment and have access to the data collected for each experiment.

In response to this challenge we propose to extend the Portal’s wiki capability to easily refer to all the relevant experiments and collected data, as well as integrate powerful analytical capability based on the R environment which will not only allow the embedding of the resulting graph, but also allow any user, to verify and even extend the analysis.

We believe that with this extensions, any LabWiki page can become an executable paper fulfilling the criteria posed in the challenge brief and we are committed to develop these extensions and demonstrate their viability within the timetable provided.

Acknowledgements

The authors would like to thank The Tung Nguyen and Didier El Baz for letting us use the P2PDC architecture as an example for the LabWiki. This work was achieved in the context of the NaDa and Onelab2 projects funded by the E.U. 7th Framework Program, and the GENI (Global Environment for Network Innovations) initiative funded by the U.S. National Science Foundation.

References