Reproducing Network Experiments With Popper

Andrea David
UC Santa Cruz
andavid@ucsc.edu

Mariette Souppe
UC Santa Cruz
msouppe@ucsc.edu

Katia Obraczka
UC Santa Cruz
katia@soe.ucsc.edu

Ivo Jimenez
UC Santa Cruz
ivo.jimenez@ucsc.edu

Sam Mansfield
UC Santa Cruz
smansfie@ucsc.edu

Kerry Veenstra
UC Santa Cruz
veenstra@ucsc.edu

ABSTRACT

In the mobile and wireless network domain, research experiments are often conducted using a variety of simulation and software development tools. Many of these simulation platforms, however, require their own operating system and specific setup, which makes replicating and validating research results a challenging task. There is a prominent need for a service that enables researchers to reproduce results with available tools in their own environment with no clashing dependencies. In this paper we use a convention client tool called Popper to make reproducibility of networking experiments less of a daunting task. In particular, we detail the steps taken to automate the execution and re-execution of experimental results presented in [**??** ,1]. Using Popper, we develop a workflow and test it on an existing experiment to compare the results from the original paper and reproduced outcome.

CCS CONCEPTS

• **Software and its engineering → Software performance**; **Software testing and debugging**; **Acceptance testing**; *Empirical software validation*; • **Social and professional topics → Automation**;

ACM Reference Format:
Andrea David, Mariette Souppe, Katia Obraczka, Ivo Jimenez, Sam Mansfield, and Kerry Veenstra. 2018. Reproducing Network Experiments With Popper. In *ICPE '18: ACM/SPEC International Conference on Performance Engineering, April 9–13, 2018, Berlin, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3184407.3184422

1 INTRODUCTION

The ability to reproduce previous experiments is one of the most important aspects in scientific research. However, as scientific discovery is rapidly advancing, researchers are pressured to rush publication of new findings and present breakthrough discoveries at conferences. Lately, there has been a growing concern in the research community about results that cannot be reproduced and thus cannot be verified. This is partially due to the lack of incentive for researchers to revoke experiments that were unable to be reproduced [2]. Moreover, replicating scientific experiments in the field of mobile and wireless networks is a challenging task. One of the biggest setbacks of reproducibility is the complexity that comes with rebuilding the same environment in which the original experiment was conducted [3]. Rerunning an original experiment can be strenuous since extensive configuration and software installation is needed in order to reproduce any experiment. In the scientific community, it is important to be able to reproduce existing research paper results to further improve upon and understand conveyed ideas.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICPE '18, April 9–13, 2018, Berlin, Germany
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5095-2/18/04…$15.00
https://doi.org/10.1145/3184407.3184422

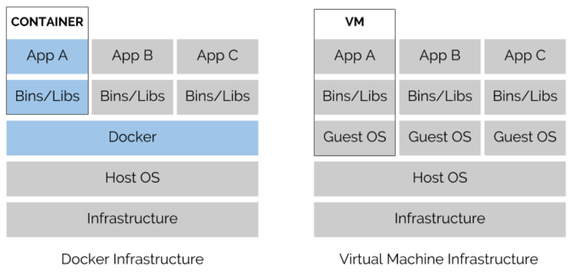


Figure 1: As you can see there is a difference between the virtual machine and the docker infrastructure. In a Docker container, it only contains the application itself and libraries for that experiment since the Docker layer takes care of the experiment environment. In a virtual machine an operating system has to first be defined before running the application.

2 RELATED WORK

2.1 Before Reproducibility Tools

Network simulators were not designed with the concept of reproducibility which makes the automation and configuration of simulations more difficult. Previously, a tool that was commonly used to rerun experiments was a virtual machine. Recently, DevOp tools, such as Docker, became popular for packaging and recreating environments as one package. The base functionality between a virtual machine and Docker are similar in the ability to create an environment for an application. However, these two tools use their own approaches. In a virtual machine you must define an environment per application, whereas in Docker an environment can be shared among other applications where the applications are run in a container. This allows for sharing resources on your host operating system since Docker dynamically allocate memory. Another added feature to Docker is that modules and packages can also be defined when building the environment which prevents having to be careful of how each operating system handles downloading modules such as *pip* for example.

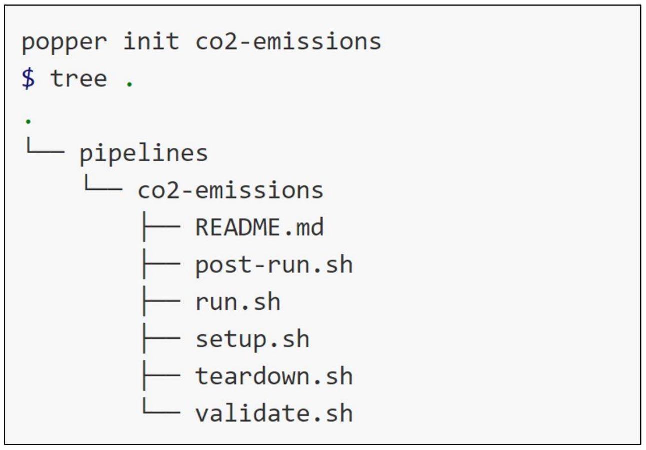
However, in order configure a virtual machine with all of the correct dependencies and environments to replicate an experiment, there is still some ambiguity in terms of the exact settings of the original experiment. Using Docker and packaging an environment with all the extra modules eliminates this ambiguity. Fig. 1 shows the infrastructure of these tools.

2.2 Exisiting Tools

Recently, researchers have started developing tools that aim to make the reproduction of scientific findings simple and undemanding. Sciunit [4] is one of such tools developed to encapsulate the author's workflow. It is a command-line tool that saves all dependencies related to executing a command in Linux. Consequently, the command can be later replicated in a different environment without having to obtain the same dependencies related to executing that command. ReproZip [5] is another tool that allows for easy reproduction of experiments using command-line executions. While executing the experiment in its original environment, ReproZip tracks operating system calls and packs necessary components of the experiment into a file. In the unpacking step, ReproZip can reproduce the experiment form that file in another environment.

2.3 Popper

Popper is a convention for creating reproducible scientific articles and experiments [6]. The convention is based on open source software (OSS) development model, using the DevOps approach to implement different stages of execution. The Popper Convention creates self-contained experiments that do not rely on libraries and dependencies other than what is already inside the "popperized" experiment. To achieve reproducibility, Popper uses pipelines that contains shell scripts that execute the original experiment. Fig. 2 shows an example Popper pipeline from one of the Popper lessons.



```
popper init co2-emissions
$ tree .
.
└── pipelines
    └── co2-emissions
        ├── README.md
        ├── post-run.sh
        ├── run.sh
        ├── setup.sh
        ├── teardown.sh
        └── validate.sh
```

Figure 2: Illustration how a general pipeline looks like using Popper

3 NETWORK SIMULATION

3.1 Background

In many network simulations, especially wireless sensor network simulations, Cooja is used as the main tool to conduct experiments. Cooja is a simulation tool for the Contiki open source operating system, which allows simulations of small and large networks.

3.1.1 TerrainLOS. One of the experiments that we reproduced in this paper is based on [Sam Mansfield's paper]. TerrainLOS is an outdoor terrain propagation model that aims to create a more accurate simulation of outdoor sensor network communication. When it comes to simulating sensor networks, terrain plays a significant role in evaluating the performance of network functions. Most simulation platforms, however, either assume a completely flat terrain or tend to use very simplistic channel propagation models that do not represent realistic outdoor terrain conditions. To present a more accurate outdoor simulation model, TerrainLOS uses common geographical height maps, called Digital Elevation Models (DEMs). These data files are used in experimental evaluations to investigate communication between nodes under realistic conditions. TerrainLOS describes the Average Cumulative Visibility (ACV) as a metric to characterize network connectivity over the terrain classified in the DEMs. ACV denotes the percentage of sensor nodes that are visible in an area from all locations on a given map. For example, 100% ACV means that every node is visible from all locations, which further implies the presence of a flat terrain. In the experimental methodology, the authors of TerrainLOS refer to the average number of nodes in an area as the average population, and the average number of nodes that a given node can communicate with is referred to as the average node density. These two metrics are used in determining network connectivity. The experiment focuses on automating the execution and re-execution of one of the experimental simulations performed in the paper, called Experimental Connectivity. This experiment focuses on experimentally evaluating the accuracy of connectivity results based on the models presented in the paper. The connectivity results are plotted using the average cumulative visibility metric and population size.

3.1.2 2.5D Deployment on TerrainLOS. The experiment where this methodology is used on is based on the paper Guiding Sensor-Node Deployment Over 2.5D Terrain [1]. The idea of this experiment is to initially place nodes on a specified terrain map within a predetermined range. Then using the algorithm from the paper, nodes continually move around the given terrain until a final cumulative visibility value has been computed. As a result, the output of the experiment is a final cvis value where of all of the nodes are placed in a way that maximizes the coverage a given terrain.

The old way the experiment [1] was conducted was that the experiment would have to start with a GUI, configure a few parameters such as the number of nodes, terrain, and transmitting range, then initial a script, run the script, and wait for final results. For each simulation these manual steps had to done.

4 POPPERIZATION

4.1 Background

4.2 TerrainLOS

To run TerrainLOS in COOJA, without using Popper, a user would have to go through the same steps I did when I tried replicating Sam's results on my computer. First, they would have to download Instant Contiki, install a Virtual Machine to start Contiki. Then, they would have to download all the necessary files and dependencies from Sam's GitHub page, create a jar file of TerrainLOS, run the simulator, and load it into COOJA. This is a very time-consuming task and the user might even receive error messages when they try to run the project the first time. Using Popper is a much more effortless way to re-execute someone's experiment without the need to have the original author explain the steps they need to take to run the project.

In my Popper pipeline, I had two stages – the run stage and the post-run stage. My setup stage was done using Docker, by creating an image of the Contiki operating system including the COOJA simulator in a Dockerfile. In the run stage, I ran Sam's python script called calc_experimental_connectivity.py that takes ACV and population as it's input. The run stage runs the experiment with population of 1, 10, 30, and 80, and ACV of 10-100 in increments of ten. The results of these runs will be put into log files. These log files are then read with the graph_max_density_connectivity.py script in the post-run stage. At the end of the run the graph of the experiment will be saved as "Connectivity.png" file under Connectivity. This 3 "popperized" experiment can be now run by just simply using the command "popper check" inside the pipeline.

4.3 2.5D Deployment on TerrainLOS

When first running the experiment [1], there were a few tools that had to be downloaded before getting the experiment to work. Java and Contiki had to be installed since those are the environments where the experiment runs. Once the environment was set up, the code for the experiment would run in Cooja. Then for every experiment to be run, a simulation file had to be configured per experiment manually.

This part of the process can be very lengthy since each simulation contains numerous different parameters. After each simulation script has been configured, each script could be run within the simulator, then after a certain amount of time the final cvis value is obtained which the part of the experiment results.

With the Popper version, there are two stages in the pipeline - the setup stage and run stage. The setup stage builds a Docker container which creates the necessary environment for the experiment to run. Additionally, the setup stage creates simulation scripts for every experiment the user would like to run. In the run stage, each of the scripts that have been made from the setup stage are now run in the Cooja simulator.

Furthermore, in the Popper version the user only has to configure one file for multiple simulations where popper will run each simulation individually and then output the final results. Workflow of automation of experiment: First, the values of the parameters of the experiment have to be defined by the user. Second, a Docker container is created with the entire environment, modules, and packages for the experiment to run. In the third step, the simulation template gets pulled in to the fourth step when creates N amount of simulations that the user has defined. Fifth, those N simulations are run and lastly the Cooja.testlog are outputted into the output folder to further evaluate the final result.

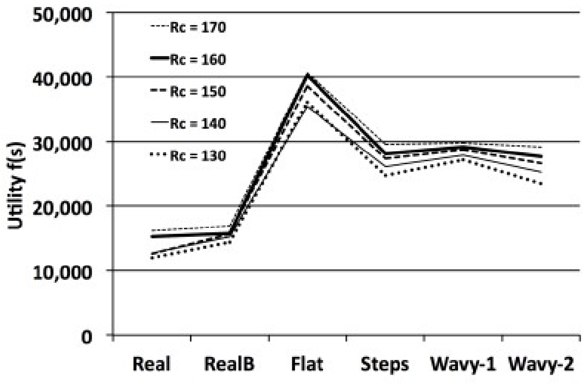


Figure 3: Original results from the 2.5D Terrain Experiment

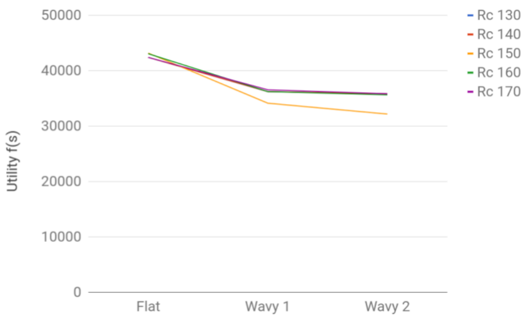


Figure 4: Reproduced results using Popper

5 RESULTS

5.1 TerrainLOS

5.2 2.5D Deployment on TerrainLOS

The final output obtained from the experiment is the final cumulative visibility value from the final placement of the nodes according to the specified terrain in the experiment.

In Fig. 3 and Fig. 4 we can see that the results are *not* exactly the same. Some of the reproduced results do not have all of the terrains as in the orginal results because not all of the terrains were available while reproducing the experiment. Furthermore, the values in Fig. 4 are higher than the values in Fig. 3. This is because the original paper was programmed in C++ and since then the experiment has been translated into a Cooja environment, which results a difference in the end result.

The main take away from these results, despite some missing elements, is that the trend of both Fig. 3 and Fig. 4 are similar.

6 CONCLUSION

Researchers usually don't have the opportunity to reach out to the author of an experiment when rebuilding their work in the environment the original project was built in. This process can be time-consuming and often impossible. The Popper convention offers a very straight forward way to implement reproducibility for scientific research articles.

It is harder to try to make a fully finished experiment reproducible, especially if the person trying to reproduce the experiment is not the original author of that experiment. Reproducibility should be kept in mind at the start of an experiment. Reproducing small sections of the code is easier than trying to reproduce code that has already been completed.

REFERENCES

[1] K. Veenstra and K. Obraczka, "Guiding sensor-node deployment over 2.5 d terrain," *Communications (icc), 2015 ieee international conference on*, IEEE, 2015, pp. 6719–6725.

[2] J. Kurose, "Dear colleague letter: Encouraging reproducibility in computing and communications research," *National Science Foundation*, Oct. 2016. Available at: https://www.nsf.gov/pubs/2017/nsf17022/nsf17022.jsp.

[3] I. Jimenez, A. Arpaci-Dusseau, R. Arpaci-Dusseau, J. Lofstead, C. Maltzahn, K. Mohror, and R. Ricci, "PopperCI: Automated reproducibility validation," *Computer communications workshops (infocom wkshps), 2017 ieee conference on*, IEEE, 2017, pp. 450–455.

[4] "Sciunit," *Sciunit*. Available at: https://sciunit.run/.

[5] "ReproZip," *ReproZip*. Available at: https://www.reprozip.org/.

[6] I. Jimenez, M. Sevilla, N. Watkins, C. Maltzahn, J. Lofstead, K. Mohror, A. Arpaci-Dusseau, and R. Arpaci-Dusseau, "The popper convention: Making reproducible systems evaluation practical," *Parallel and distributed processing symposium workshops (ipdpsw), 2017 ieee international*, IEEE, 2017, pp. 1561–1570.