Internet measurements on EdgeNet

Kevin Vermeulen, Burim Ljuma, Olivier Fourmaux, Timur Friedman, Rick Mcgeer

Internet measurements on EdgeNet

Kevin Vermeulen *Sorbonne Université*
Burim Ljuma *Sorbonne Université*
Olivier Fourmaux *Sorbonne Université*
Timur Friedman *Sorbonne Université*
Rick McGeer *US Ignite*

***Abstract*—We describe the deployment of an Internet measurement experiment to three testbeds that offer Linux containers hosted at widely distributed vantage points: the well-established PlanetLab Central and PlanetLab Europe platforms, and the new EdgeNet platform. The experiment results were published in the proceedings of ACM IMC 2018. We compare the capabilities of each testbed and their effect on the ease of deployment of the experiment. Because the software for this experiment has several library dependencies and requires a recent compiler, it was easiest to deploy on EdgeNet, which is based on Docker and Kubernetes. This extended abstract is accompanied by a demonstration of the reproducible deployment of a measurement tool on EdgeNet.**

***Index Terms*—internet measurements, testbeds**

I. Introduction

We describe the deployment of an Internet measurement experiment to three testbeds that offer Linux containers hosted at widely distributed vantage points: the well-established PlanetLab Central (PLC) [3], [9] and PlanetLab Europe (PLE) [4], [10] platforms, and the new EdgeNet [1], [7], [8] platform; we operate the latter two, in collaboration with other partners. The experiment consisted in testing the new Multilevel MDA-Lite Paris Traceroute (MMLPT) tool and the results were published in the proceedings of ACM IMC 2018 [12]. We compare the capabilities of each testbed and their effect on the ease of deployment of the experiment. This extended abstract is accompanied by a demonstration of the reproducible deployment of a measurement tool on EdgeNet available at `https://gitlab.planet-lab.eu/cartography`.

II. Experiment Overview

A. Context

Traceroute is a tool that can be run from a host in the Internet in order to gather a sequence of Internet Protocol (IP) addresses corresponding to the routers that are traversed by packets as they follow a path towards a specified destination. Created by Van Jacobson in 1988 [11], variants of the tool have been proposed and adopted by the networking research community, including Paris Traceroute [6], which enables tracing of the multi-path routes that are enabled by load balancing routers. Multilevel MDA-Lite Paris Traceroute (MMLPT) [12] is a recent improvement that aims to reduce the original's probing overhead while also revealing *aliases*, or IP addresses belonging to the same router. We have contributed to both of these tools, and, in our recent evaluation of the latter, we aimed to use vantage points provided by PLC, PLE, and EdgeNet.

TABLE I
Testbed Comparison

	PLC	PLE	EdgeNet
Container Technology	Linux-VServer (2002–2014) LXC (2014–present)		Docker proprietary
Image Management	None		Docker
Host machine	bare metal* (*VM at one PLE site)		bare metal or VM
Image OSes	Fedora 8, CentOS 6.4	Fedora 23, 24, and 25	any available Docker image
Default compiler	`gcc` 4.1, `gcc` 4.4	`gcc` 5.3, 6.3, and 6.4	depends on the Docker image
Build	within container on each node		on server of one's choice
Orchestration	manual or via an orchestrator from the research community		Kubernetes

B. Technical requirements

In the interest of rapid development, we wrote MMLPT in Python 3, making use of several libraries. To craft, send, and sniff packets, MMLPT uses the `scapy` library [5]. In order to keep track of the multi-path routes that it discovers, it uses the `graph-tool` [2] library. This consists of a Python wrapper around a C++ library that invokes the metaprogramming module of the well known C++ Boost library. To be able to compile `graph-tool`, one must have a compiler that supports C++14. These characteristics of the experiment—its use of several software libraries and its requirement for a recent compiler—are an obstacle to ease of deployment on distributed platforms. Both the Debian and Ubuntu Linux distributions provide a precompiled `graph-tool` package, which can ease deployment when one of those environments is available.

III. Testbeds

PLC, the original PlanetLab, created in 2002, with sites mainly in North America and Asia; PLE, which came online in 2008, and is predominantly present in Europe; and EdgeNet, created in 2018, with a combination of North American and European sites, are each slicable thanks to a host container technology that supports lightweight Linux containers. The initial PlanetLab container technology was Linux-VServer, and then PLC and PLE jointly made the switch to LXC. There is no image management system for PLC and PLE, whereas EdgeNet uses Docker. Table I summarizes these technical characteristics. There are fewer sites with active nodes today than at PlanetLab's peak in 2011, as researchers tend to maintain their nodes while they are conducting work that

requires the platform, but no longer have the resources to look after them afterwards. Even reduced in size, the ability that the platform offers of being able to deploy software on nodes around the world remains valuable, as the regular sign-ups of new institutions to PLE testifies. Since Docker can easily be installed in a VM, we look to EdgeNet as a technology that will be easier to set up and maintain than the dedicated servers that PlanetLab requires. With the exception of one PLE site that made a special effort to install nodes in VMs, all of the PlanetLab hosts are bare metal.

IV. Comparative testbed experiences

A. Build

As mentioned, our MMLPT tool depends upon the `graph-tool` library. The first step in building the tool is to compile the library. We did this successfully on PLE and EdgeNet, but not on PLC due to the older image OSes on that platform, which do not support C++14. We tried installing a newer version of `gcc` on PLC nodes, but gave up because the new `gcc` wouldn't compile on the old `gcc`; presuming it was possible, we would have had to upgrade `gcc` in stages.

Compiling `graph-tool` was resource intensive, both in terms of CPU and RAM, due to the `graph-tool` library's extensive use of template metaprogramming. On our test server, with an Intel Xeon X5660 2.80GHz, 6 core CPU, and 48 GB RAM, it took a mean, over ten trials, of 3 hours to get MMLPT installed with all of its dependencies. This presents a challenge to existing PlanetLab systems, as LXC does not accept externally generated images in the way that Docker does. We tried, but failed, to package a binary of MMLPT for Fedora that we could ship to PLE nodes. The remaining option was to compile MMLPT on each PLE node. As resources in a container on a shared machine tend to be scarcer than those on a dedicated server, compilation tended to take longer than on our test server. Worse, compilation did not always succeed on these remote nodes. Of the 40 PLE nodes available during our study, we succeeded in installing MMLPT with all of its dependencies on 25 of them.

Because EdgeNet uses Docker for container image management, there is no need to build on a remote server to a narrowly specific OS supported by that server. It is possible to install a wide range of Linux OSes on an EdgeNet node. This allowed us to avoid fastidious compilation by choosing an OS for which a `graph-tool` package already exists. Between Debian and Ubuntu, we chose Ubuntu 18.04, and it took just a few minutes to build an image on our test server.

B. Deploy

As we built MMLPT directly on each PLE node that supported this, there was no additional deployment step necessary. On EdgeNet, one must push the Docker image to a Docker repository, then send a YAML file containing a pointer to this image to the head node of the Kubernetes instance that provides EdgeNet orchestration. The head node then deploys the image to one's chosen nodes. It took a few minutes for our Docker image to be deployed to EdgeNet's 29 nodes.

C. Run

With MMLPT installed and ready to be used on PLE and EdgeNet, we needed to pilot our measurements. For PLE, we used a Python program with the Paramiko library to access each node and execute remote commands. We then collected the results via the `scp` module of Paramiko on our server, for analysis. On EdgeNet, we again used a Python program, but instead of accessing the nodes via `ssh`, we used the `ssh`-like access provided by Kubernetes itself. Then, we collected the data from the EdgeNet nodes via the `scp`-like command line interface of Kubernetes.

V. Conclusion

This extended abstract describes how we have run Internet measurements on PlanetLab and EdgeNet. It illustrates some difficulties we encountered with older operating systems and remote compilation. Our work provides a proof of concept that motivates us, in our role as testbed operator, to migrate PlanetLab Europe (PLE) to the EdgeNet software. Based on modern Docker and Kubernetes technologies, we believe that EdgeNet will facilitate the work of researchers who turn to PLE to run experiments on distributed nodes. Because EdgeNet nodes can be installed on VMs and not just bare metal machines, we believe that this will allow PLE to grow and better maintain its installed base of nodes over time.

Acknowledgments

A research grant from the French Ministry of Defense has made this work possible. K. Vermeulen, B. Ljuma, O. Fourmaux, and T. Friedman are associated with Sorbonne Université, CNRS, LIP6, F-75005 Paris, France. K. Vermeulen and T. Friedman are associated with the LINCS, F-75013 Paris, France.

References

[1] EdgeNet. http://edge-net.org.
[2] graph-tool. https://graph-tool.skewed.de.
[3] PlanetLab. https://www.planet-lab.org.
[4] PlanetLab Europe. https://www.planet-lab.eu.
[5] Scapy. https://scapy.net.
[6] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with Paris Traceroute. In *Proc. ACM IMC*, 2006.
[7] M. Berman, T. Friedman, A. Gosain, K. Keahey, R. McGeer, I. Moerman, A. Nakao, L. Nussbaum, K. Rauschenbach, V. Syrotiuk, et al. Report of the third global experimentation for future internet (GEFI 2018) workshop. *arXiv preprint arXiv:1901.02929*, 2019.
[8] J. Cappos, M. Hemmings, R. McGeer, A. Rafetseder, and G. Ricart. EdgeNet: A global cloud that spreads by local action. In *Proc. IEEE/ACM Symposium on Edge Computing (SEC)*, 2018.
[9] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.
[10] S. Fdida, T. Friedman, and T. Parmentelat. OneLab: An open federated facility for experimentally driven future internet research. In *New Network Architectures*, pages 141–152. Springer, 2010.
[11] V. Jacobson. 4BSD routing diagnostic tool available for ftp. Email 8812201313.AA03127@helios.ee.lbl.gov to the IETF and end2end-interest e-mail lists, 1988.
[12] K. Vermeulen, S. D. Strowes, O. Fourmaux, and T. Friedman. Multilevel MDA-Lite Paris Traceroute. In *Proc. ACM IMC*, 2018.

VI. Demo

Our demo shows how to run Internet measurements on EdgeNet, using the example of the Multilevel MDA-Lite Paris Traceroute (MMLPT) tool described in this extended abstract. The entire sequence, from sign-up to data collection, takes about 10 minutes. It requires an Internet connection. We will encourage people to come with their own laptops and sign up for EdgeNet, leaving with a fully working example experiment to serve as a template for their future work. Those who do not bring a laptop will be able to watch as we deploy the experiment on our own computer.

A. Sign-up

First of all, we sign up a new user of the EdgeNet platform by accessing the URL: `https://sundewcluster.appspot.com`. Then, once the account has been validated, we download the configuration file using this URL: `https://sundewcluster.appspot.com/download_config`.

B. Write and push the Docker image

```
FROM ubuntu:bionic

RUN apt-get update
RUN apt-get -y install vim
RUN echo "deb http://downloads.skewed.de/apt/bionic bionic universe" >> /etc/apt/sources.list
RUN echo "deb-src http://downloads.skewed.de/apt/bionic bionic universe" >> /etc/apt/sources.list
RUN apt-get -y install gnupg
RUN apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 7A80C8ED4FCCBE09
RUN apt-get update
RUN apt-get -y install git
RUN apt-get -y install python-pip

RUN pip install scapy numpy scipy netifaces requests

RUN DEBIAN_FRONTEND=noninteractive apt-get install -y python-graph-tool

RUN git clone https://gitlab.planet-lab.eu/cartography/multilevel-mda-lite.git
RUN python multilevel-mda-lite/setup.py install
```

Fig. 1. Our Docker file containing instructions to install Multilevel MDA-Lite Paris Traceroute.

Our Docker file, shown in Fig. 1, contains the Docker instructions to build an image containing MMLPT and all its dependencies. Once the Docker file is correct and tested, we push the image to the Docker repositories via the command `Docker push <username>/<image_name>`.

C. Deploy the Docker image

Once we have the configuration file and the image pushed to the Docker repository, two options are possible for uploading our image to the cluster: the Kubernetes command line interface, and the interactive dashboard.

1) Command Line Interface: The configuration file must be placed at the `$HOME/.kube/config` location. This is where Kubernetes looks for the configuration file by default. Then we create a YAML file. Our example YAML file is shown in Fig. 2. The `apiVersion` field is given by the EdgeNet tutorial. The `kind` field specifies the type of deployment that you want on the EdgeNet pods, typically the number of nodes on which the Docker image will be deployed, and the number of instances of the image per node. The `spec:hostNetwork` field specifies that we want to be able to open some ports of our container. As we want to run some multilevel multipath traceroutes, we will open network sockets to be able to receive the ICMP packets.

```
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: multilevel-mda-lite-paris-traceroute
spec:
  template:
    metadata:
      labels:
        app: multilevel-mda-lite-paris-traceroute
    spec:
      hostNetwork: true
      containers:
        - name: multilevel-mda-lite-paris-traceroute
          image: bljuma/multilevel-mda-lite-paris-traceroute
          command: [ "/bin/bash", "-c", "--" ]
          args: [ "while true; do sleep 30; done;" ]
```

Fig. 2. Example of YAML file pointing to our Docker image.

Therefore, this parameter is set to `true`. Then we specify the name of our deployment in `spec:containers:name` and the corresponding Docker image that we previously pushed to the Docker repository. Here, we have chosen the name `multilevel-mda-lite-paris-traceroute` and the image is our username followed by the name of our deployment. To make the container run forever, we add the two fields `spec:containers:command` and `spec:containers:args`. Without these two fields, the container is just instantiated and executed, and it returns.

D. Run measurements and collect data

Now that MMLPT is installed on the EdgeNet infrastructure, we will launch some traceroutes from the EdgeNet nodes towards some destinations and gather the results on our local machine. This system is orchestrated by our Python script, that uses the Kubernetes command line interface. First, we need to obtain the list of the hashes used to map our container to each EdgeNet physical node. For this to be done, we use the command: `kubectl get pods`. Then, after having extracted the mapping, for each container we launch the kubectl exec command: `kubectl exec -ti <hash> --bash -c "python multilevel_paris_traceroute.py <options>"`. Finally, via `kubectl cp hash:<path/to/the/output/> <path/to/our/data/directory>` we collect the generated data onto our machine. We can reuse a `kubectl exec` command to clean up the environment when needed.