Distributed Network Experiment Emulation

Giuseppe di Lena*†, Andrea Tomassilli*, Damien Saucez*, Frédéric Giroire*,
Thierry Turletti*, Chidung Lac†, and Walid Dabbous*
*Université Côte d'Azur, Inria, CNRS, France
†Orange Labs, France

***Abstract*—With the ever growing complexity of networks, researchers have to rely on test-beds to be able to fully assess the quality of their propositions. In the meanwhile, Mininet offers a simple yet powerful API, the goldilocks of network emulators. We advocate that the Mininet API is the right level of abstraction for network experiments. Unfortunately it is designed to be run on a single machine. To address this issue we developed a distributed version of Mininet – Distrinet – that can be used to perform network experiments in any Linux-based testbeds, either public or private. To properly use testbed resources and avoid over-commitment that would lead to inaccurate results, Distrinet uses optimization techniques that determine how to orchestrate the experiments within the testbed. Its programmatic approach, its ability to work on various testbeds, and its optimal management of resources make Distrinet a key element to reproducible research.**

I. Introduction

Emulation is an important step in the evaluation cycle of network applications and protocols. It provides a fully controlled and reproducible environment to test real protocols/applications in a reproducible way, without any modification. Mininet is the reference emulator in the network community but it has been designed to run on a single machine only. To address this limitation, we propose an extension of Mininet, called Distrinet, which allows running Mininet scenarios by distributing them on any pool of computing resources including Linux cluster of machines (e.g., Grid5000 [2] or any Linux-based testbed such as R2lab [1]) or the cloud (e.g., Amazon EC2).

Position statement. It is interesting for testbed operators to provide network emulation capabilities in their testbeds in order to run large scale scenarios with realistic and accurate results. The Mininet API offers the right level of abstraction to achieve this goal and Distrinet is the way to implement it in testbeds. Benefiting of such abstraction is likely to attract new users to the testbeds, particularly those interested in the mix of emulation and real experimentation approaches. At the same time, Distrinet can also be used by testbed operators to automate networking tasks. Using Distrinet on federated platforms such as Fed4Fire testbeds provides a uniform interface for running emulation scenarios including hybrid ones that involve more than one testbed.

Related Work. To distribute experiments Maxinet [8] runs multiple instances of Mininet, one per cluster node. From that point of view, Maxinet philosophy is similar to ours. However, instead of extending Mininet, it uses it and therefore does not provide the exact same API as Mininet, resulting in potential incompatibilities. Containernet [4] is an extension of Mininet that isolates nodes using docker containers and can be combined with Maxinet to distribute network experiments. Distrinet combines the concepts of Maxinet and Containernet in such a way that it is fully compatible with Mininet, offering so the ability to prototype experiments with Mininet then to deploy them at scale on a testbed with Distrinet.

Contributions. Distrinet shares the Mininet API such that Mininet programs and scripts can readily be used for experiments involving testbeds. The development of Distrinet led to the following key contributions:

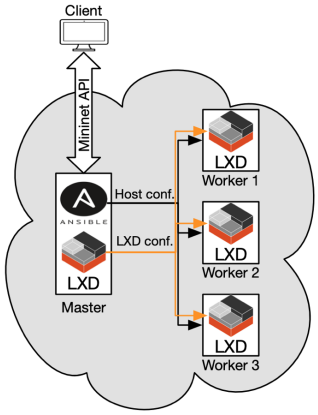
- **Architecture. (Sec. II)** We have extended the implementation of Mininet to enable it to support intensive compute and I/O experiments on a large variety of experimental infrastructures (e.g., a single computer, a Linux-based testbed, or even the Amazon EC2 cloud). Our solution, called Distrinet relies on Ansible and LXD to transparently orchestrate distributed experiments.
- **Optimization. (Sec. III)** We propose optimization techniques to optimally allocate resources to the experiments. Given that information on the experimental infrastructure may be partial or not, we defined two distinct optimization strategies. On the one hand, if the infrastructure network topology is unknown (e.g., in public clouds such as Amazon EC2) we use a Vector Bin Packing with Multiple-Choice optimization scheme. On the other hand, if the experimental infrastructure topology is known (e.g., in private testbeds or clusters) we use Virtual Network Embedding (VNE) techniques to best allocate resources.

II. Architecture

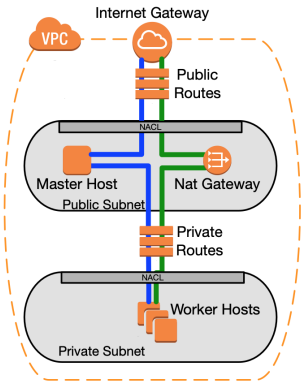
Fig. 1a shows the high level architecture of Distrinet that is used to distribute a network experiment on multiple machines.

Mininet emulates nodes with user-level processes and isolates them with Linux cgroups and network namespaces. Distrinet moves forward and uses LXC containers to properly isolate resources and LXD to simplify the management of complex scenarios. To connect nodes together, network links are emulated with virtual Ethernet logically connected with VxLAN tunnels. Network conditions such as link rate, losses, and delay are directly inherited from Mininet.

In addition to orchestrating experiments, Distrinet offers an infrastructure provisioning mechanism. This mechanism relies on Ansible to install and configure LXD, SSH, and package dependencies on each machine taking part in the experiment. When the experiment runs on Amazon EC2, Distrinet instantiates first a Virtual Private Cloud (VPC) configured as depicted in Fig. 1b where the virtual instances running the experiment will be deployed. It also determines the required number of machines to use by solving an optimization



(a) General architecture



(b) AWS VPC configuration.

problem (see Sec. III for more details). If the experiment is performed in Grid5000 [2], hosts are automatically booked and configured to support the experiment. Once the infrastructure is provisioned, Distrinet optimally determines the placement of hosts, switches, and links in the infrastructure, as explained in Sec. III. You can simply extend Distrinet by creating a Provision class for your test-bed. There are few methods to implement to make it compatible with your test-bed; the user can use the grid5000 Provision class as a guideline to implement its own class.

III. Resource allocation

The virtual network can be modelled with an undirected graph with node (e.g., CPU, memory) and link (e.g., bandwidth) requirements. Basically, we identified two different cases:

- with full control of the testbed infrastructure
- with a partial control of the infrastructure (e.g, Amazon EC2)

These two use cases lead to the following distinct problems.
The Virtual Network Embedding problem. In this case we have full control on the resources, so we can optimally allocate them. The objective consists in minimizing the number of machines to run an experiment. The VNE problem is known to be extremely hard, even for finding a feasible solution. Rost et al. [5] show that the problem of finding a feasible embedding is NP-Complete, even for a single request. We propose three heuristics that provide near-optimal solutions in a reasonable computation time: (i) *k-balanced* carries out a partition of logical nodes, while minimizing the network capacity needed between physical nodes. We use as a subroutine an algorithm for the k-balanced partitioning problem given in [7]; (ii) *Greedy* first builds a bisection tree, where each node is divided into 2 partitions of nodes. Then, the tasks are assigned to the available physical machines by doing a bread first search visit on the tree, assigning to each machine the first fitting subtree; (iii) *DivideSwap* divides the logical nodes into n physical machines and, then, iteratively swap them to reduce the network cut weight.

Vector Bin Packing with Multiple-Choice (VBP-MC). When experiments run in a public cloud, the problem consists in choosing a set of virtual machine instances taken from a set of instance types, which provide different combinations of CPU, memory, disk, and networking. In this case, the natural objective consists in minimizing the cost to run an experiment. This problem is often referred to as the *Vector Bin Packing with Multiple-Choice*. The three following heuristics approaches have been implemented in Distrinet: *First Fit Weighted Sum* [6], *Best Fit Dop Product* [6], and *First Fit Ordered Deviation* [3].

IV. Conclusion

We proposed Distrinet that extends Mininet to distribute network experiments on multiple machines in Linux-based testbeds or cloud. Distrinet computes the optimal number of hosts required for an experiment. In the case of Amazon EC2, Distrinet automatically launches the required number or instances and groups them within a VPC. In the case of Grid5000, Distrinet automatically books and deploys machines in the desired Grid5000 cluster. This automatic provisioning of the experimental infrastructure allows to hide the details of the experimental infrastructure to the user that can then focus on the actual experiment, regardless of the infrastructure that is used. Distrinet implementation currently supports deployment on Linux hosts, Grid5000, and Amazon EC2 but it has been conceived to make it trivial to port to another experimental infrastructure (e.g., R2lab). Thanks to its programmatic approach and its flexible design, Distrinet is the perfect candidate to offer a uniformed interface to running scenarios that involve more than one testbed (e.g., the federated Fed4Fire testbeds).

V. Collaborations

Distrinet source code is publicly available on https://github.com/Giuseppe1992/Distrinet and anyone can benefit from it. This project is made in collaboration between Inria, CNRS, Université Côte d'Azur, and Orange labs.

References

[1] R2lab testbed. In *https://r2lab.inria.fr*.
[2] D. Balouek, A. Carpen Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, L. Nussbaum, O. Richard, C. Pérez, F. Quesnel, C. Rohr, and L. Sarzyniec. Adding virtualization capabilities to the Grid'5000 testbed. In I. I. Ivanov, M. van Sinderen, F. Leymann, and T. Shan, editors, *Cloud Computing and Services Science*, volume 367 of *Communications in Computer and Information Science*, pages 3–20. Springer International Publishing, 2013.
[3] B. T. Han, G. Diehr, and J. S. Cook. Multiple-type, two-dimensional bin packing problems: Applications and algorithms. *Annals of Operations Research*, 50(1):239–261, 1994.
[4] M. Peuster, H. Karl, and S. van Rossem. Medicine: Rapid prototyping of production-ready network services in multi-pop environments. In *IEEE NFV-SDN*, pages 148–153, Nov 2016.
[5] M. Rost and S. Schmid. Charting the Complexity Landscape of Virtual Network Embeddings. In *Proc. IFIP Networking*, 2018.
[6] P. Silva, C. Perez, and F. Desprez. Efficient heuristics for placing large-scale distributed applications on multiple clouds. In *ACM CCGrid*, 2016.
[7] H. D. Simon and S.-H. Teng. How good is recursive bisection? *SIAM Journal on Scientific Computing*, 18(5):1436–1445, 1997.
[8] P. Wette, M. Dräxler, A. Schwabe, F. Wallaschek, M. H. Zahraee, and H. Karl. Maxinet: Distributed emulation of software-defined networks. In *2014 IFIP Networking Conference*, pages 1–9, June 2014.