Demo: The Hangar environment for Teaching and Research in Programmable Networking

Nik Sultana
Illinois Institute of Technology
Department of Computer Science
Chicago, USA

***Abstract*—This demo presents Hangar, a VM-based environment that grew out of components that evolved over the last few years to support research and teaching into programmable networking.**

In addition to showing Hangar, the demo will discuss use-cases and requirements that were encountered when developing Hangar and its predecessors. Hangar is designed to provide an environment that is easy to update and use. It includes fully-scripted VM generation, packages frequently-used tools and minimizes dependencies, and provides a suite of examples.

***Index Terms*—Virtualization, Programmable Networking, Computer Science Education, P4**

I. Introduction

Development, testing, evaluation and demonstration environments have several independent components, which can make them difficult to set up and maintain. These parts consist of packages and tools whose versions and mutual compatibility change over time.

Hangar is a VM-based environment that evolved over the last four years from an enabler for research [1], [2] to become a platform for teaching [3], [4], and now serves for both teaching and research in programmable networking. It provides wrapping and glue intended to reduce the effort of maintaining, updating, and using the VM. Using Hangar, students can be up and running in minutes. New versions of the VM can be generated more easily to keep up with new versions of components.

The purpose of this demo is to present Hangar to the wider community, share experience with other researchers and educators, and gain feedback.

II. Background

Hangar started out as a collection of scripts that were used to run the Wharf [1] prototype for link-layer FEC. These scripts were significantly extended during the Flightplan [2] project, the development and testing of which required flexible composition of virtual network elements through simple and evolvable configurations. This work also involved customizing the instantiation of Mininet [5] to run scripted experiments.

These scripts were then evolved further to form the so-called "back-end" of FDP [3], which repurposed those scripts to support a teaching platform that provided in-browser visualizations.

978-1-6654-8234-9/22/$31.00 ©2022 IEEE

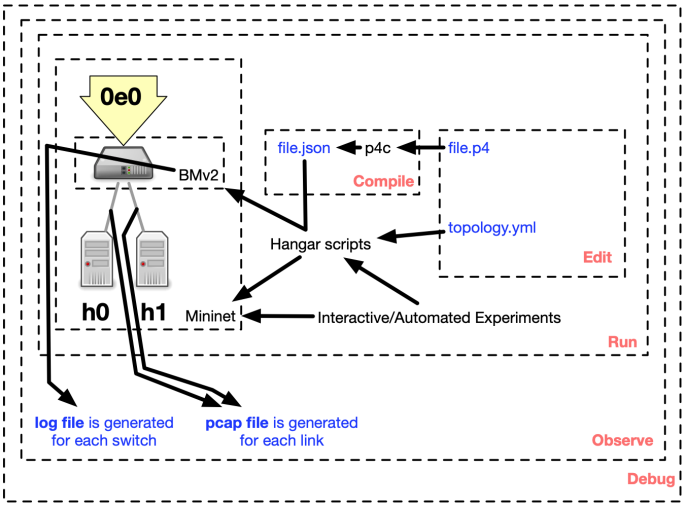

Fig. 1. Workflow of using Hangar showing input and output files.

Hangar took its present form when it was adapted for use in the CS595 course on "Designing Large-Scale Networked Systems" [4] taught at Illinois Institute of Technology during Spring semester 2022. Until that point, the scripts forming Hangar had dependencies and references that made them difficult to disentangle from Flightplan.

In addition to being disentangled from Flightplan, Hangar also provides new usability features that make it more convenient to use. As well as being used to teach CS595, Hangar was also used by students to carry out individual research projects [6] at Illinois Tech. To help students get started, Hangar packages a set of tools and documents how to use them in the workflow shown in Fig. 1.

III. Examples

The demo will cover how to define and configure a virtual programmable network from scratch. This is based on an example used in IIT's CS595 course. A screenshot from this example is shown in Fig. 2, which shows Hangar interacting with Mininet during the initialization, configuration, and execution stages of an experiment.

A second example involves simulating a fat-tree network [7]. This example is much more complex. It features a topology generator that was developed to evaluate the artefacts

Fig. 2. Worked example being given during CS595 [4].

in the Flightplan [2] project, which have since been released as open source under a permissive license.

IV. Usage Workflow

Hangar consists of three workflows. One workflow is followed by end-users of the VM. The other two workflows are carried out by the creators or maintainers of a VM—such as a researcher, teaching assistant, or course instructor.

This section focuses on the workflow followed by end-users, which is shown in Fig. 1. It consists of the following stages: Edit, Compile, Run, Observe, and Debug. Each stage can involve providing or inspecting files, and using various tools.

The filesystem in Hangar's VM contains a directory that is structured into the following directories:

- `networks/` contains example topologies, source code, and experiment scripts.
- `BMv2/` contains supporting scripts. Users will usually invoke these indirectly, and need not inspect them.
- `doc/` contains documentation that describes the YAML encoding for topology and configuration.
- `lib/` contains supporting P4 code that is used by several examples, and that can be used by end-users to define their own switch behavior. It's often useful for users to inspect these files.
- `build/` is auto-generated when users compile P4 code. A generic Makefile is provided to support the building of P4 projects in Hangar.

To help users keep track of the data that is generated during each run of Hangar, a directory called `hangergames_output/` is automatically created in their current directory when an experiment is run. It contains logs and packet captures related to that experiment.

V. Related Work

There are other virtualized networking environments for research prototyping and teaching. Instead of being used through Hangar, Mininet [5] can be used directly but would require customization to achieve similar features. Hangar provides those features by using Mininet's API and through additional layers of external wrapping, to streamline configuration of virtual networks that contain several programmable elements. Another notable example is the VM that is used for P4 tutorials [8], which offers similar usability benefits. In comparison, Hangar (1) opts for the simpler CLI-based configuration of the dataplane instead of using P4Runtime, (2) has a slightly more featureful configuration interface, and (3) Hangar's VM generation is fully scripted.

Acknowledgment

I thank the contributors to the Flightplan and FDP projects, and in particular Isaac Pedisich, John Sonchack, Heena Nagda, and Rakesh Nagda for their contributions to what became Hangar. I am also indebted to Luis Casarrubios Elez, Cyprien Gueyraud, Gonzalo Ignacio Hidalgo Garcia, Pratul Palaniappa Muthuraja, Shivam Patel, and Xue Zhang for their feedback on the use of Hangar during the CS595 course at Illinois Tech during Spring 2022.

References

[1] H. Giesen, L. Shi, J. Sonchack, A. Chelluri, N. Prabhu, N. Sultana, L. A. Kant, A. J. McAuley, A. Poylisher, A. DeHon, and B. T. Loo, "In-network computing to the rescue of faulty links," in *Proceedings of the 2018 Morning Workshop on In-Network Computing, NetCompute@SIGCOMM 2018, Budapest, Hungary, August 20, 2018*, X. Jin and C. Kim, Eds. ACM, 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1145/3229591.3229595

[2] N. Sultana, J. Sonchack, H. Giesen, I. Pedisich, Z. Han, N. Shyamkumar, S. Burad, A. DeHon, and B. T. Loo, "Flightplan: Dataplane Disaggregation and Placement for P4 Programs," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. USENIX Association, Apr. 2021, pp. 571–592. [Online]. Available: https://www.usenix.org/conference/nsdi21/presentation/sultana

[3] H. Nagda, R. Nagda, S. Sheth, N. Sultana, and B. T. Loo, "FDP: A Teaching and Demonstration Platform for Networking," in *SIGCSE '21: The 52nd ACM Technical Symposium on Computer Science Education, Virtual Event, USA, March 13-20, 2021*, M. Sherriff, L. D. Merkle, P. A. Cutter, A. E. Monge, and J. Sheard, Eds. ACM, 2021, p. 1376. [Online]. Available: https://doi.org/10.1145/3408877.3439543

[4] N. Sultana, "CS595 Designing Large-Scale Networked Systems (Spring 2022)," http://www.cs.iit.edu/~nsultana1/teaching/S22CS595/.

[5] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX. New York, NY, USA: Association for Computing Machinery, 2010. [Online]. Available: https://doi.org/10.1145/1868447.1868466

[6] N. Sultana, "Opportunities for Student Research Projects," http://www.cs.iit.edu/~nsultana1/student_projects/.

[7] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, p. 63–74, aug 2008. [Online]. Available: https://doi.org/10.1145/1402946.1402967

[8] P. L. Consortium, "P4 Tutorial," https://github.com/p4lang/tutorials.