Simple and efficient ICN network virtualization with vICN

Jordan Augé*, Giovanna Carofiglio*, Marcel Enguehard*†,
Luca Muscariello*, Mauro Sardara*†
*Cisco Systems †Telecom ParisTech

ABSTRACT

Information-Centric Networking (ICN) has been proposed as an alternative to IP for future networks such as 5G. To speed up its development and adoption, researchers and engineers require testing tools that are both simple and scalable. In particular, it is crucial to be able to quickly deploy ICN-enabled network topologies in a flexible and efficient manner.

In this demonstration, we showcase vICN (virtualized ICN), a platform that enables easy deployment, orchestration and management of ICN networks. vICN uses standard virtualization technologies such as Linux Containers (LXC) and is fully integrated with the CICN suite to enable flexible testing of ICN technologies on general-purpose hardware. Furthermore, it can perform live monitoring and modification of the network. In particular, we use vICN to deploy a simple topology that consists of 9 nodes. We show that vICN bootstraps the topology in about 60s on commodity hardware. We then demonstrate how vICN interacts with the virtualized network and how it can be used for easy experimentation.

CCS CONCEPTS

• **Networks → Network experimentation**; **Network management**; *Programmable networks*; *Network monitoring*;

KEYWORDS

Deployment; Virtualization; Management; Measurements

ACM Reference format:
Jordan Augé, Giovanna Carofiglio, Marcel Enguehard, Luca Muscariello, and Mauro Sardara. 2017. Simple and efficient ICN network virtualization with vICN. In *Proceedings of ICN '17, Berlin, Germany, September 26–28, 2017,* 2 pages.
https://doi.org/10.1145/3125719.3132100

1 INTRODUCTION

ICN has been identified as a candidate for future network architectures such as 5G [1]. However, ICN must show its maturity and prove concrete advantages over competing state-of-the-art technologies in real networks. With that in mind, some authors have emphasized the need for an experimental-based research approach based on application development [2]. In particular, ICN would benefit from experiments at scale with production-ready code.

A quick glance at recent work on ICN reveals three approaches to experimental research: (i) experiments on production-like networks; (ii) simulation (e.g., with ndnSIM [4] or ccnSim [3]); (iii) testbeds (e.g., the NDN and CUTEi [5] testbeds). Experiments on realistic networks sound appealing, but they require heavy human and financial investment to realize at scale. Moreover, they provide low-flexibility, in particular for testing new-generation networks that involve a lot of mobility. Simulation tools offers important characteristics for efficient testing: they are flexible in terms of network design, tested applications, and run on commodity hardware. However, they are not accurate enough to provide more than preliminary clues w.r.t. a system behaviour. Testbeds allow users to run production code on an actual network and are thus more realistic. Yet they do not offer the same flexibility as simulation does. For instance, ICN testbeds have physical limits (e.g., number of nodes in testbed) which are not necessarily in sync with the need of users. Furthermore, they are bound by the technology choices of their creators, which constrains the experimental space (e.g., choice of a specific ICN technology). We thus see the need for a tool combining the flexibility of simulators with the realism of testbeds.

In this demonstration, we present vICN (virtualized ICN), a tool for deployment, orchestration and management of ICN networks. It uses standard virtualization technologies to create virtual ICN networks. It is designed to provide the flexibility of simulation with the realism of production networks. While vICN can run on commodity hardware, it can also be used as an instrumentation and orchestration tool for deploying experiments on a existing testbed. vICN thus complements current approaches, paving the road towards flexible and simple experiments with ICN. The code source of vICN is open-source and available as part of the Linux-Foundation FD.io project [7]. In particular, our demonstration can be reproduced in a few minutes by anyone with a Linux machine.

2 VICN DESCRIPTION

We represent a typical vICN deployment in fig. 1. It is composed of Linux Containers, linked together through virtual Ethernet interfaces (veth). On top of the underlying topology (i.e., virtual nodes and links), vICN implements its own SDN controller, tailored to set up ICN (over Ethernet, TCP, or UDP) and/or IP addressing and routing. It also sets up an IP-based control network using a virtual bridge, thus giving internet connectivity to the containers. That connectivity is used to install missing applications and report monitoring information about the network. In the containers, relevant ICN application are started and linked with the ICN forwarder.

Similarly to the ONOS Intent Framework [6], vICN allows user to declare a topology in a declarative (rather than imperative) fashion. Using object relation mappings between the various network

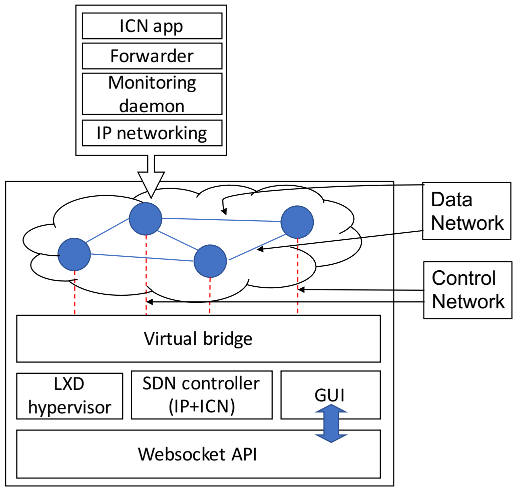

Figure 1: Blocks deployed by vICN

resources and finite-state machines to track their state, vICN is thus able to deploy a full ICN network from a minimal expressed intent. The theory and general ideas behind vICN are detailed in [9].

In this paper, our contribution is to demonstrate the current state of vICN and its potential as a unifying tool for deploying and managing ICN networks. In particular, vICN exploits existing virtualization tools while shielding the user from their complexity. The wished network deployment is declared through a JSON file. The user only needs to described the networks he wants as well as the position of ICN producers. For instance, our demonstration which consists of 9 nodes, can be expressed in a JSON file of 137 lines. It is also possible to declare a network using a Python API (similarly to MiniCCNx [8]), to ensure that the network is fully programmable.

3 DEMONSTRATION

Our demonstration can be decomposed in three parts. First, we bootstrap a network using vICN that consists of three ASes with three nodes each. We then show how we can monitor network activity using our GUI while creating traffic with the CICN suite. Finally, we show how we can use vICN to modify the network online by adding and removing nodes in the topology. Our demonstration thus showcases the following key features of vICN:

1) **Fast network deployment:** using the model described in [9], vICN is able to quickly deploy a fully functioning network from a 200 lines JSON file. In particular, vICN (i) creates the virtual topology (nodes and links), (ii) sets up the IP network (addressing and routes), (iii) sets up the ICN network (forwarders and routes), and (iv) starts ICN applications when asked. Thanks to a multithreaded scheduler, it deploys our 9 nodes topology in just above a minute.

2) **Realistic conditions:** vICN runs real production code on its virtualized network. In particular, it is fully integrated with the rest of the open-source CICN software-suite. As such, vICN is particularly fit for advanced experiments before deployment on real networks. In our demonstration, we show how we test specific features of the CICN Metis forwarder such as per-packet load-balancing using vICN. Furthermore, vICN is able to mimic real-life

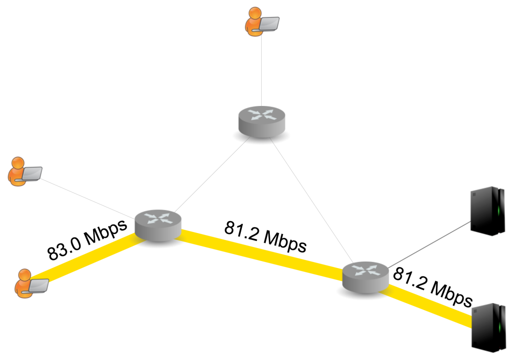

Figure 2: GUI monitoring of the demonstration topology

constraints, such as constrained link bandwidth using features of the Linux kernel.

3) **Live monitoring:** vICN reports live telemetry on the network, such as resource state or link bandwidth. It uses the API described in [9] to retrieve that information, and displays it on a web interface. This telemetry can be used to monitor the system, as well as to gather statistics and information about its performance. We show an example of the interface in fig. 2.

4) **Live topology changes:** vICN offers a python API through which the topology can be manipulated in real time. In our demonstration, we show that we can seamlessly add, remove or move nodes in the virtualized network with minimal overhead. This feature is especially important to conduct experiences on dynamic scenarios such as 5G or IoT deployments.

4 CONCLUSION

Through this demonstration, we illustrate the capacities of vICN and how it helps foster ICN research. Through its agile framework, it allows researchers to quickly bootstrap ICN networks while focusing on their specific problematic. Furthermore, it is efficient enough to enable fast prototyping or serialized experiments without much overhead.

ACKNOWLEDGMENTS

This work benefited from support of NewNet@Paris, Cisco's Chair "Networks for the Future" at Telecom ParisTech (https://newnet.telecom-paristech.fr)

REFERENCES

[1] ITU-T Recommendation ITU-T Y.3071 Data Aware Networking (information centric networking): Requirements and capabilities. *In ITU Study Group 13 Final Rep.*, 2017
[2] L. Zhang et al. Named Data Networking. *In Comput. Commun. Rev.*, 2014
[3] M. Tortelli et al. A Hybrid Methodology for the Performance Evaluation of Internet-scale Cache Networks. *In Elsevier Comput. Networks*, 2017
[4] A. Afanasyev et al. ndnSIM: NDN simulator for NS-3, *Tech. Rep. NDN-0005*, 2012
[5] H. Asaeda et al. Container-based unified testbed for information-centric networking. *In IEEE Network*, 2014
[6] Onos Intent Framework https://wiki.onosproject.org/display/ONOS/Intent+Framework [Consulted Jul. 2017]
[7] The Linux Foundation The Fast Datapath Project (FD.io) https://fd.io
[8] C. Cabral et al. Mini-CCNx: fast prototyping for named data networking *In Proc. ACM ICN*, 2013
[9] M. Sardara et al. Virtualized ICN (vICN): towards a unified network virtualization framework for ICN experimentation. *In Proc. ACM ICN*, 2017