Reproducing Real NDN Experiments using Mini-CCNx

Carlos M. S. Cabral, Christian Esteve Rothenberg, Maurício Ferreira Magalhães
Faculty of Electrical and Computer Engineering (FEEC)
University of Campinas (UNICAMP), São Paulo, Brazil
{cabral,chesteve,mauricio}@dca.fee.unicamp.br

ABSTRACT

This demo presents Mini-CCNx as a new experimentation tool for the NDN (Named Data Networking) model. Rooted in recent container-based emulation and resource isolation techniques developed to foster research in SDN/OpenFlow, Mini-CCNx features a number of contributions to support and facilitate NDN experiments using the project's official code base at scale. In addition to integrating all the available pieces of code, the platform offers experimenter-friendly configuration interfaces tailored to NDN and allows to run arbitrary topologies with hundreds of nodes without sacrificing high-fidelity results. We demonstrate how Mini-CCNx is capable of reproducing experiments on the NDN testbed using dynamic routing protocols (OSPFN) and multicast content delivery (NDNVideo). We import the whole NDN testbed topology including annotated links and show how the obtained results match published results. The experience suggests that Mini-CCNx can be a helpful experimental platform prior to going to a real deployment, altogether reducing time and costs, and yielding early insights on the end application behavior, routing configuration needs, caching characteristics, protocol performance, and so on.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network communications

Keywords

ICN, NDN, CCN, emulation, prototyping, routing

1. INTRODUCTION

Experimentally driven research in Information-Centric Networking (ICN) is crucial to the evaluation of open research issues in this area, such as routing protocols, forwarding strategies, caching techniques, content-centric application development and so on.

Ideally, it would be interesting to have a testing and development platform with characteristics such as *(i) flexibility* – it should be possible to rapidly create several scenarios using different configuration parameters, *(ii) scalability* –it should be possible to create topologies with a sufficiently high number of nodes, *(iii) low-cost* –the platform should be able to run in a commodity laptop/desktop or single Amazon EC2 instance, and *(iv) realism* –the behavior should be similar to a real deployment, the code executed within the tool should be the same as the one on real hardware/testbeds, and the system should be able to generate and receive real traffic, e.g., from the Internet or the local network.

ICN'13, August 12, 2013, Hong Kong, China.
ACM 978-1-4503-2179-2/13/08.

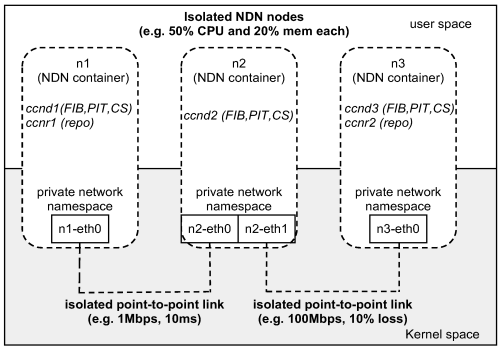

Figure 1: Three NDN nodes connected using Mini-CCNx containers

Towards these goals and filling an existing gap in the available research platforms (mainly simulators and testbeds), Mini-CCNx [1] appears as a low-cost, scalable and experimenter-friendly NDN emulator (hundreds of NDN nodes can be instantiated in a commodity laptop), with flexible options to define experiment configurations, including topologies, link properties, and applications. Furthermore, Mini-CCNx runs real code with real traffic and presents high-fidelity results with regards to real experiments.

2. MINI-CCNx OVERVIEW

Mini-CCNx is a fork of Mininet-HiFi [3] (originally proposed for OpenFlow/SDN) augmented with several mechanisms, tools, and classes for NDN experimentation. Each node emulates a NDN node and runs the official code [1].

Mini-CCNx uses Container-Based Emulation (CBE) [4], a lightweight OS-level virtualization technique. Each container allows groups of processes to have independent views of system resources, such as process IDs, file systems and network interfaces while still using the same kernel. Each container is a NDN node, with its own network namespace,

[1]Available at https://github.com/carlosmscabral/mn-ccnx

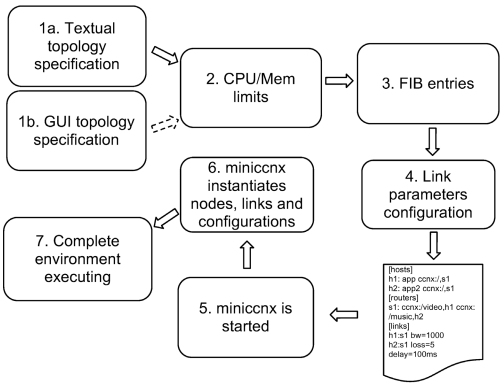

Figure 2: Typical Mini-CCNx workflow

virtual network interface(s), NDN-specific data structures implemented by the `ccnd` daemon (PIT, FIB, and CS) and repositories (`ccnr`). These nodes are connected to each other using virtual Ethernet links in the kernel space. The main challenge Mini-CCNx faces is related to performance isolation and so it uses isolation techniques in order to limit the resources available for each node and link and thus providing high-fidelity results. In order to achieve this, Linux `cgroups` [2] are used to limit CPU bandwidth for each node. Mini-CCNx also adds memory limits, a relevant subject for NDN when it comes to caching and content storage. Finally, using `tc`, it is possible to configure several link properties such as bandwidth, delay, and packet loss. Figure 1 illustrates the CBE and isolation features used by Mini-CCNx.

3. DEMONSTRATION

The demonstration is split in two parts, each following a typical Mini-CCNx experimental workflow (see Fig. 2).

1. Retrieving content under different network conditions. The experiment shows how the download time of media files in NDN repositories is affected under different link conditions such as low-latency LAN, high-delay links, and lossy links (e.g. wireless links). Download performance depends also on hop distance and the caching configuration. With this, we want to show how easy it is to set-up different NDN scenarios using Mini-CCNx. Figure 3 shows a Mini-CCNx running environment.

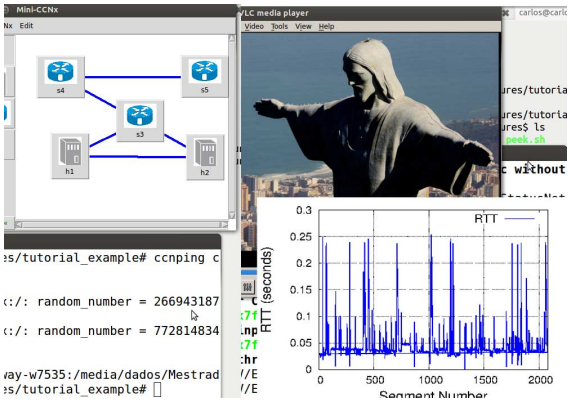

Figure 3: Mini-CCNx demo experiment including GUI (upper left), CLI (bottom left), NDNVideo (upper right), and measured RTT (bottom right).

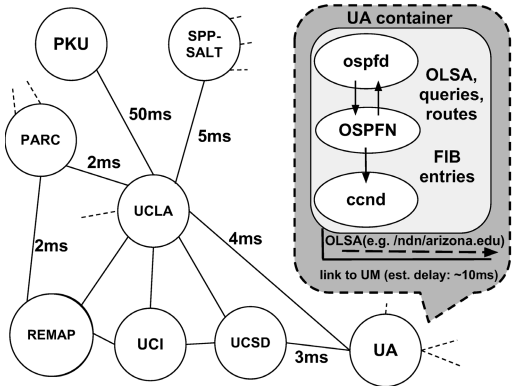

Figure 4: Part of the emulated NDN Testbed (out of scale) and approximated link delays

2. Reproducing the NDN Testbed. Emulating the NDN testbed [6] with Mini-CCNx, we evaluate the behaviour of (1) multicast NDNVideo [7] delivery, and (2) the `ospfn` [8] dynamic routing protocol . With a simple `miniccnx -testbed` command, Mini-CCNx automatically loads 17 nodes representing the NDN testbed routers and their point-to-point links.[2]. Propagation delays were estimated using straight geographical distances between the nodes (see Fig. 4).

2.1 Multicast content delivery. NDNVideo [7] is a video streaming application that allows to play live and pre-recorded videos with random access and no session negotiation. We will show how Mini-CCNx reproduces the unexpected behavior seen on the real deployment (RTT spikes), and also how the suggested solution indeed attenuated this problem (cf. TR-NDN-0007 [5]).

2.2 Dynamic routing. In this scenario, each node runs its own instance of the `ospfn` [8] daemon, and the required `ospfd` and `zebra` daemons from Quagga [9]. Each node is configured to announce exactly the same name prefixes outlined in the NDN testbed website [6]. For instance, the CAIDA/UCSD router announces the `/ndn/caida.org/` prefix while the UA router announces `/ndn/arizona.edu`. The user can verify the routing scheme convergence and the timing and insertions of FIB entries (as seen with `ccndstatus`). Moreover, with Mini-CCNx, the user can easily bring any link up or down and observe the dynamic behavior of OSPFN such as convergence times and automatic failover.

4. REFERENCES

[1] CCNx. CCN implementation. `https://www.ccnx.org/`.
[2] cgroups. Linux Control Groups. `https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt`.
[3] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown. Reproducible network experiments using container-based emulation. *CoNEXT '12*, page 253, 2012.
[4] lxc. Linux Containers. `http://lxc.sourceforge.net/`.
[5] NDN Project. NDN Technical Reports. `http://www.named-data.net/techreports.html`.
[6] NDN Testbed. NDN Routing Topology. `http://netlab.cs.memphis.edu/script/htm/topology.html`.
[7] NDNVideo. `http://www.named-data.net/techreport/TR007-streaming.pdf`.
[8] OSPFN. OSPF for Named-data. `http://www.named-data.net/techreport/TR003-OSPFN.pdf`.
[9] Quagga. Quagga Routing Suite. `http://www.nongnu.org/quagga/`.

[2]As seen on March 8th 2013 in [6]