Demo Abstract: Realistic Evaluation of Kernel Protocols and Software Defined Wireless Networks with DCE/ns-3

Emilio P. Mancini, Hardik Soni
Thierry Turletti, Walid Dabbous
INRIA, FRANCE
{emilio.mancini, hardik.soni,
thierry.turletti, walid.dabbous}@inria.fr

Hajime Tazaki
University of Tokyo
Japan
tazaki@wide.ad.jp

ABSTRACT

We propose to demonstrate Direct Code Execution (DCE), a framework that enables to execute nearly unmodified applications and Linux Kernel code jointly with the ns-3 simulator. DCE allows therefore fully deterministic reproducibility of network experiments. DCE also supports larger scale scenarios than real-time emulators by using simulation time dilatation. In this demonstration, we will showcase two main scenarios: (1) a basic example describing how to integrate in DCE the Data Center TCP (DCTCP) Linux kernel patch, and then how to customize this protocol and run it on different scenarios; (2) a more advanced use case demonstrating how to benefit from DCE to build a rich and realistic evaluation environment for Software Defined Wireless Networks based on Open vSwitch and the NOX SDN controller.

Categories and Subject Descriptors

I.1.6.7 [**Simulation and Modeling**]: Simulation Support Systems

Keywords

Direct Code Execution; Emulation; Linux; Network Stack; Simulation

1. INTRODUCTION

The need for reproducible realistic and large scale network experiments has been highlighted for several years by the networking community. Reproducibility is essential for researchers and developers in order to give them the opportunity to verify and possibly enhance proposals made by other researchers but also, it helps debugging the code. Today, two complementary approaches are widely used to evaluate the performance of networking protocols: simulators and emulators. Recently, Direct Code Execution (DCE) [1,6,11] has been proposed. This framework benefits from the pros of both simulation and emulation, while limiting the main drawbacks of the two approaches. To the best of our knowledge, DCE is the only free open source framework that enables to evaluate real implementations of network systems in a scalable and reproducible manner, while allowing easy debugging within a deterministic and reproducible environment. In particular, DCE can run unmodified network systems written in C/C++ on top of the ns-3 networking simulator. DCE uses the ns-3 simulated time clock which makes it more scalable than most of emulators such as Mininet [8] by relaxing the real time execution constraint. In this way, CPU-greedy scenarios will take longer than they would in real time, but the results obtained will be accurate.

Basically, DCE is structured around three layers, as shown in Figure 1. DCE dynamically links the applications with the POSIX layer. It is built upon the core and kernel layers to re-implement the standard socket APIs used by emulated applications. Not the entire set of POSIX API is currently supported, so new applications may require to extend this layer to add possible missing system calls. The kernel layer takes advantage of the core services to provide an execution environment to the Linux network stack within the single-processed network simulator. The services of kernel such as the Linux bottom halves, scheduler, and timer API are presented as a new architecture based on the asm-generic implementation to minimize the modifications of the original kernel code. The core layer, at the lowest-level, handles the virtualization of stacks, heaps, and global memory. It uses a single-process model that executes every simulated process within the same host process and isolates the namespace of each simulated process. Further information on DCE is available in [6,11].

At the last MSWiM'13 conference in Barcelona, we successfully demonstrated DCE with two use cases: 1) seamless handover involving a Linux MPTCP implementation and 2) information-centric networking over ad-hoc networks based on the CCNx stack from PARC [10]. The feedback we obtained was very fruitful, stressing the benefits of demonstrating DCE with scenarios of different complexity levels, and studying its scalability. In this new demo we will showcase two new scenarios now made possible with DCE: a use case showing how to integrate the Data Center TCP (DCTCP) protocol [5] to a real kernel stack, and a more advanced use case demonstrating a wireless mesh scenario that involves the NOX [7] Software Defined Networking (SDN) controller and Open vSwitches [9]. The former use case is a simple example that will allow to demonstrate how easy it is to customize a network protocol. In particular, we will explain how to use a patched Linux kernel to analyze the behavior of

MSWiM'14, September 21–26, 2014, Montreal, QC, Canada.
Copyright 2014 ACM 978-1-4503-3030-5/14/09 ...$15.00.
http://dx.doi.org/10.1145/2641798.2655182.

DCTCP with a real application in a simulated network. The latter scenario will explain how to conduct routing experiments in a realistic wireless environment and how to control them through the NOX SDN controller.

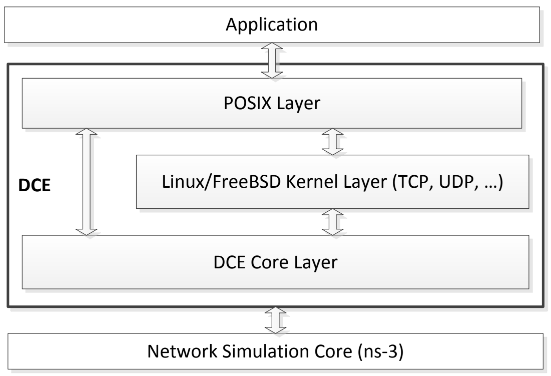
Application
POSIX Layer
DCE
Linux/FreeBSD Kernel Layer (TCP, UDP, ...)
DCE Core Layer
Network Simulation Core (ns-3)

Figure 1: DCE architecture.

2. DEMONSTRATION DETAIL

2.1 Kernel protocols and scalability

This first use case is a simple example based on DCTCP [5], an enhancement of the TCP congestion control algorithm for environments specific to data centers, i.e., involving very high bandwidth links, low round-trip times, and small buffer switches. DCTCP is provided as a Linux kernel patch to Linux 2.6.38.3.

In this experiment, we will demonstrate how to use a kernel patch to study its behavior in a simulated network with full reproducibility. Moreover, we will study the scalability of DCE and will show the advantages of using the ns-3 simulation clock instead of the wall clock. The experiment will be run with different number of senders and a single receiver running the well-known *iperf* application, as shown in Figure 2.

Figure 3 shows a snapshot of the simulation animation using the NetAnim [2] tool that will be used during the demo, where we see packets flowing from twenty senders to the queue monitor.

Figure 4 plots the wall clock time required to simulate 5 seconds of TCP data transmission in function of the number of nodes. By using the ns-3 simulation clock, DCE is able to run CPU-greedy simulations scenarios that can not be run in real time on the machine due to its limited capacities.

2.2 Simulating Software Defined Wireless Networks with DCE

Testing, evaluating applications or carrying out research experiments involving a software defined wireless networking architecture requires a simulation tool, capable of generating network performance metrics and results for any required scenario, in a repeatable way. Keeping these requirements in mind, in this second use case, we plan to demonstrate the execution of the OpenFlow controller NOX [7] and Open vSwitch [3] on DCE/ns-3 to simulate a software defined wire-

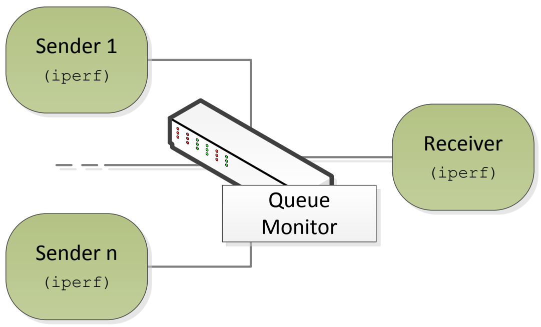

Figure 2: Schema of the DCTCP experiment.

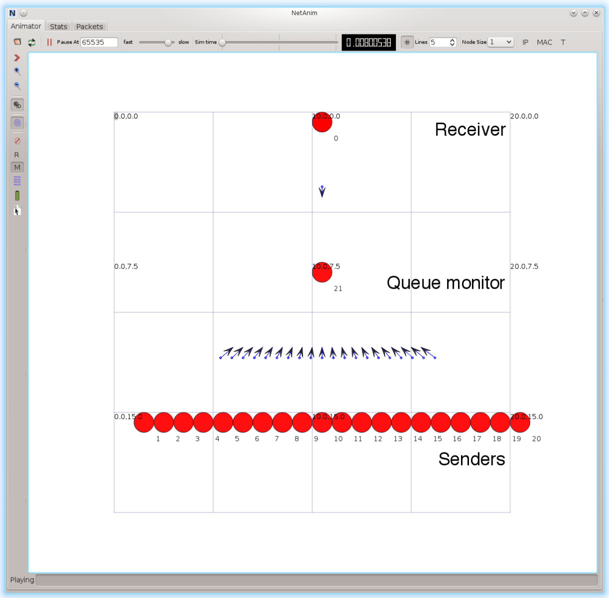

Figure 3: Snapshot of the animation showing the evolution of the simulation using the NetAnim tool.

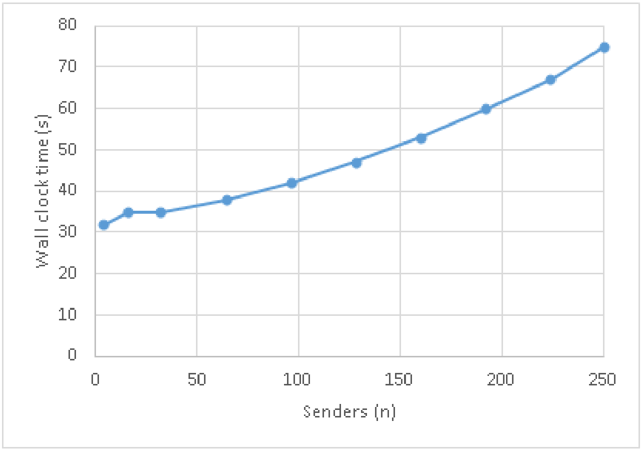

Figure 4: Wall clock time to simulate a 5s transmission for different numbers of nodes.

less network. Note that such a wireless environment is not possible to be emulated in a realistic way using Mininet [8] because the latter can only emulate point-to-point physical links using virtual ethernet pairs (e.g., MAC layer is ignored), and it does not provide mobility models for wireless nodes. Meanwhile, DCE can take advantage of different mobility models available in ns-3 along with various MAC layer protocols. In this demo, we will demonstrate the simulation capabilities available for research on SDN in a wireless context.

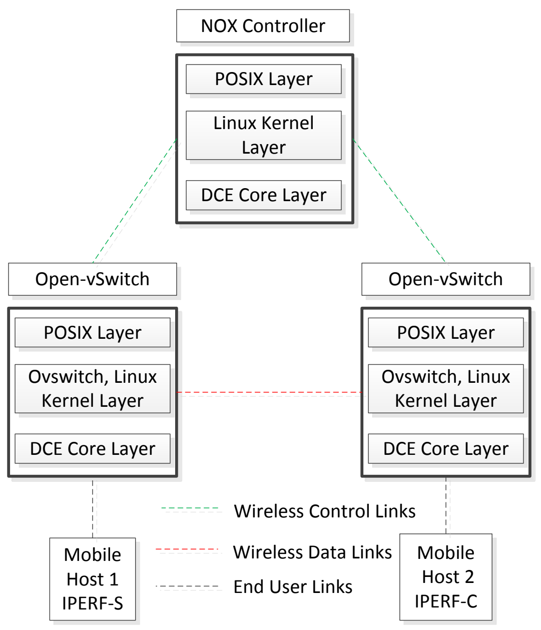

Figure 5: Simulation of a software defined wireless network

Figure 5 shows an architectural view of our proposed simulation environment. The Control Plane (CP) of the SDN architecture can be simulated using a different physical channel than the Data Plane (DP) as shown in the figure in red and green colour links, respectively. NOX code is executed on a simulated ns-3 node using DCE to simulate a SDN controller. OpenFlow-enabled wireless routers are simulated using the Open vSwitch distribution. Note that Open vSwitch can be used with or without kernel support. Purely user-space based configuration is considered experimental and not thoroughly tested as mentioned in [4]. We use Open vSwitch with data-path kernel module support as it is widely used. DCE provides a mechanism to incorporate such a kernel module based application execution, in the same way it is done for the kernel-space linux implementation of MPTCP demonstrated in [10].

The proposed simulation environment is scalable enough to model a large number of wireless end users and routers, as it operates using the ns-3 simulation clock. It can reproduce experiment results without careful resource provisioning of the host machine as required in [8]. We plan to present the demo of Open vSwitch working as basic forwarding wireless router programmed by NOX and to test the connectivity using iperf running on two simulated hosts.

3. ACKNOWLEDGMENTS

This research is partially supported by INRIA and Japanese Society for the Promotion of Science (JSPS) Joint Research Projects in the context of the Simulbed/Designet *associated team*.

4. REFERENCES

[1] Direct Code Execution. www.nsnam.org/docs/dce/manual/html. [Accessed 6/20/2014].

[2] NetAnim. www.nsnam.org/wiki/NetAnim. [Accessed 6/20/2014].

[3] Open vSwitch. www.openvswitch.org/. [Accessed 6/20/2014].

[4] Open vSwitch Support. http://openvswitch.org/support/. [Accessed 6/20/2014].

[5] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data center TCP (DCTCP). *SIGCOMM Comput. Comm. Rev.*, 41(4), 2010.

[6] D. Camara, H. Tazaki, E. Mancini, M. Lacage, T. Turletti, and W. Dabbous. DCE: Test the real code of your protocols and applications over simulated networks. *IEEE Comm. Mag.*, 52(3):104–110, March 2014.

[7] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. Nox: Towards an operating system for networks. *SIGCOMM Comput. Commun. Rev.*, 38(3):105–110, July 2008.

[8] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown. Reproducible network experiments using container-based emulation. In *Proc. of CoNEXT*, pages 253–264, New York, NY, USA, 2012. ACM.

[9] J. Pettit, J. Gross, B. Pfaff, M. Casado, and S. Crosby. Virtual switching in an era of advanced edges, 2010. 2nd Workshop on Data Center. Converged and Virtual Ethernet Switching (DC-CAVES).

[10] H. Tazaki, E. Mancini, D. Camara, T. Turletti, and W. Dabbous. DCE: Increase Simulation Realism Using Unmodified Real Implementations. In *Proc. of MSWiM (demo abstract)*, pages 29–32, NY, USA, 2013.

[11] H. Tazaki, F. Urbani, E. Mancini, M. Lacage, D. Camara, T. Turletti, and W. Dabbous. Direct Code Execution: Revisiting library OS architecture for reproducible network experiments. In *Proc. of CoNEXT*, pages 217–228, NY, USA, 2013. ACM.