Direct Code Execution: Realistic Protocol Simulation with Running Code

Hajime Tazaki, Emilio Mancini, Daniel Camara, Thierry Turletti, Walid Dabbous

Direct Code Execution: Realistic Protocol Simulation with Running Code

Hajime Tazaki*, Emilio Mancini°, Daniel Câmara°, Thierry Turletti°, Walid Dabbous°
*NICT, Japan °INRIA, France

Abstract

We propose the demonstration of Direct Code Execution (DCE), a framework of network simulation running with both existing Linux kernel space protocol stack and POSIX socket based protocol implementations, to achieve a set of requirements for reproducible network experiment: 1) experimentation realism, 2) topology flexibility, 3) easy and low cost replication, 4) experimentation scalability, and 5) easy debugging. Our demonstration showcases the typical use cases of DCE: content centric networking over mobile ad hoc network with CCNx, and seamless handoff experiment with Linux the Multipath TCP implementation.

Keywords: Network Stack; Experiment; Software Development; Direct Code Execution; Linux

1. INTRODUCTION

Increasing demand for the reproducible network experiments requires sophisticated tools to conduct arbitrary network experiment with satisfying a set of requirements such as 1) experimentation realism, 2) topology flexibility, 3) easy and low cost replication, 4) experimentation scalability, and 5) easy debugging. Container-based emulation (CBE) [3] as well as shared emulation testbed (PlanetLab [9]) are good at satisfying such requirements while network simulators are not as it had been lacked the functional realism of simulated protocols.

Porting existing network protocol implementations to network simulators is one possible direction to improve the functional realism of experimental result. OppBSD [2] or INETQuagga [1] would take this approach to reuse existing protocol implementations (i.e., TCP/IP stack of FreeBSD and Quagga routing protocol suite), but they still left the painful task of manual patching for a particular network simulator, resulting difficulties to track the latest version of code. Network Simulation Cradle (NSC) [6] introduces a nice way with automatically generating C source files of different operating system's network stacks (e.g., FreeBSD, Linux, OpenBSD, lwip) built to shared libraries used in network simulators. The automation alleviates the cost of tracking latest version of codes and supports a wide range of existing code with a single framework, but it still requires additional effort to introduce arbitrary protocols implementation rather than TCP.

Table 1: Requirements for reproducible network experiments.

	Simulators	Testbeds	Emulators
Functional Realism	???	✓	✓
Timing Realism	✓	✓	✓
Topology Flexibility	✓	(limited)	✓
Easy Replication	✓	✓	✓
Easy Debug	✓		
Scalability	✓		✓

We will present the Direct Code Execution (DCE) environment for `ns-3`, notable for being the first free, open source framework for integrating both Linux kernel space protocol stack and POSIX socket based user space application code with a leading discrete-event network simulator. DCE takes the traditional library operating system (LibOS) approach such as Exokernel [7] in its core architectural design to enable running and evaluating real network protocol implementations. As a result, DCE brings us functional realism to network simulation-based experiment as shown in Table 1.

2. SYSTEM OVERVIEW

The design of DCE is structured around three separate components as depicted in Figure 1.

- **Core.** The lowest-level *core* module handles the virtualization of stacks, heaps, and global memory. It provides *singe-process model* virtualization for simulated nodes with carefully

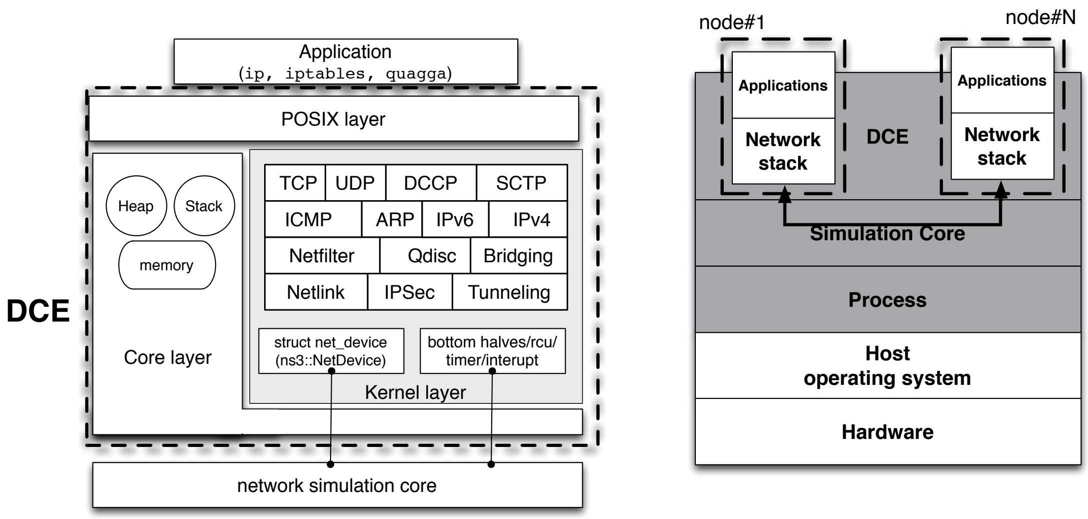


Figure 1: Architecture of Direct Code Execution.

isolating the namespace of each simulated process.

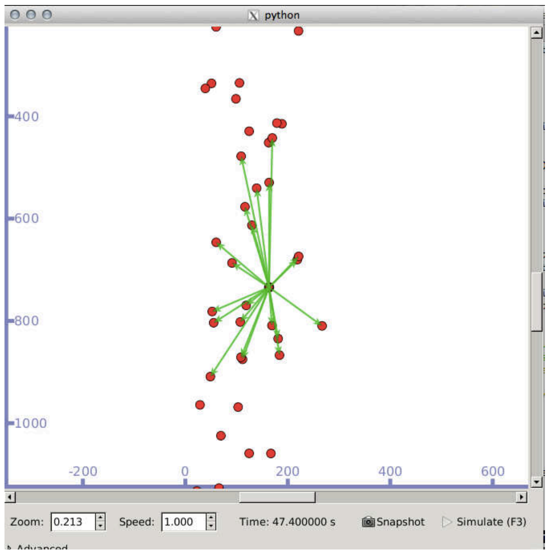
- **Kernel layer.** The kernel layer takes advantage of the *core* services to provide an execution environment to the Linux network stack within the network simulator. The services of kernel such as the Linux *bottom halves*, scheduler, Read-Copy-Update (RCU) [?], and timer API are re-implemented as a new architecture based on the `asm-generic` implementation to minimize the modification to original kernel code.

- **POSIX layer.** The POSIX layer builds upon the *core* and *kernel* layers to re-implement the standard socket APIs used by emulated applications.

Along with these components, DCE, in theory, enables to run any existing network protocol implementations upon `ns-3` without any manual modifications to the original code. At the present moment, it supports a broader range of existing implementations running on `ns-3`: `Linux kernel` (2.6.36, 3.4-3.10 version), `quagga` (ripd, ripngd, ospfd, ospf6d, bgpd, and rtadv), `ccnx`, `iperf`, `ip`, `ping/ping6`, `umip`, `bind9`, `unbound`, `thttpd`, and bittorrent (`opentracker/rasterbar`).

3. DEMONSTRATION DETAIL

We will demonstrate the seamless simulation examples using existing protocol implementations over `ns-3`. To present major features of DCE, we pick



Zoom: 0.213 Speed: 1.000 Time: 47.400000 s

Figure 2: CCNx in mobile and Wi-Fi ad hoc network.

two examples as representatives of user space application simulation and Linux kernel space protocol simulation.

User space protocol implementation running on DCE: CCNx[1] over mobile nodes

Contents Centric Network (CCN) [5] is a network architecture categorized clean-slate design, which brings a different perspective for the identifier of communications from traditional IP addresses to *named data*. Such floating identifier independent from physical conditions is beneficial especially in

[1]`http://www.ccnx.org/`

highly dynamic network topology like mobile ad hoc network [8].

In this showcase, we will present this new network paradigm with a CCN implementation, CCNx, over simulated dynamic topology via `ns-3`.

Kernel space protocol implementation running on DCE: Multipath TCP

Multipath TCP (MPTCP) [10] is an extension of the standard TCP that allows to use multiple subflows with different IP addresses without modifying user space applications. Basically, this new transport protocol makes it possible to increase the throughput of an application by running it over multiple links, as well as transparent handoff using multiple IP addresses.

In this showcase, we will present a Linux MPTCP implementation[2] running on DCE over `ns-3` with the support of various user space applications (`quagga`, `ip` command, `udhcpd`, `iperf`). Multiple addresses to a mobile node are provided via two different wireless technologies of `ns-3`, LTE and Wi-Fi, and tries to switch its primary address between them during node movement, keeping ongoing TCP session available. Similar handoff scenario using Linux Mobile IPv6 implementation is available[3].

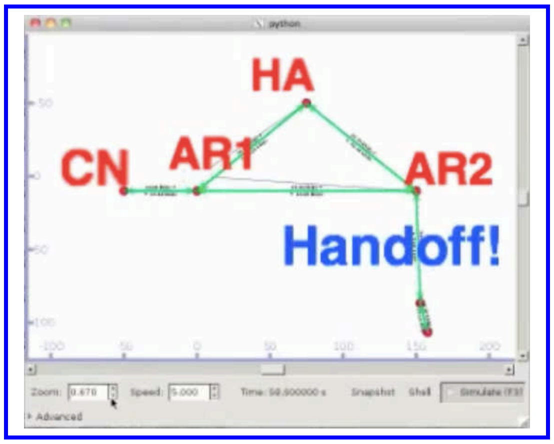


Figure 3: Similar Handoff simulation with the Linux Mobile IPv6 implementation.

In all demonstrations, we will present a typical network simulation using existing protocol implementations, with animated visualization of simulated nodes, traffic status, as well as measurement results with plotted graphs from each simulation.

Facilities for the Demonstration

The followings are the required facilities at the venue to demonstrate our system.

- Power outlet (2 slots for 2 Laptop PCs)
- Table (enough space to put 2 PCs)
- Wall or Stand to put a poster

4. REFERENCES

[1] INETQuagga: OMNeT++ wiki. `http://www.omnetpp.org/pmwiki/index.php?n=Main.INETQuagga`. (Accessed July 1st 2013).

[2] Bless, R., and Doll, M. Integration of the freebsd tcp/ip-stack into the discrete event simulator omnet++. In *Simulation Conference, 2004. Proceedings of the 2004 Winter* (2004), vol. 2, pp. 1556–1561 vol.2.

[3] Handigol, N., Heller, B., Jeyakumar, V., Lantz, B., and McKeown, N. Reproducible network experiments using container based emulation. In *Proceedings of the 2012 ACM CoNEXT conference* (2012), CoNEXT '12.

[4] Hibler, M., Ricci, R., Stoller, L., Duerig, J., Guruprasad, S., Stack, T., Webb, K., and Lepreau, J. Large-scale virtualization in the emulab network testbed. In *USENIX 2008 Annual Technical Conference on Annual Technical Conference* (2008), pp. 113–128.

[5] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies* (Dec. 2009), CoNEXT '09, ACM, pp. 1–12.

[6] Jansen, S., and McGregor, A. Simulation with real world network stacks. In *Proceedings of the 37th conference on Winter simulation* (2005), WSC '05, Winter Simulation Conference, pp. 2454–2463.

[7] Kaashoek, M. F., Engler, D. R., Ganger, G. R., Briceño, H. M., Hunt, R., Mazières, D., Pinckney, T., Grimm, R., Jannotti, J., and Mackenzie, K. Application performance and flexibility on exokernel systems. In *Proceedings of the sixteenth ACM symposium on Operating systems principles* (New York, NY, USA, 1997), SOSP '97, ACM, pp. 52–65.

[8] Meisel, M., Pappas, V., and Zhang, L. Ad hoc Networking via Named Data. In *Proceedings of the fifth ACM international workshop on Mobility in the evolving internet*

[2]`https://github.com/multipath-tcp/mptcp`
[3]`https://www.youtube.com/watch?v=y790NE3EPCg`

architecture (2010), MobiArch '10, ACM, pp. 3–8.

[9] Peterson, L., Anderson, T., Culler, D., and Roscoe, T. A blueprint for introducing disruptive technology into the Internet. *SIGCOMM Comput. Commun. Rev. 33*, 1 (2003), 59–64.

[10] Raiciu, C., Paasch, C., Barre, S., Ford, A., Honda, M., Duchene, F., Bonaventure, O., and Handley, M. How hard can it be? designing and implementing a deployable multipath tcp. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation* (Berkeley, CA, USA, 2012), NSDI'12, USENIX Association, pp. 29–29.