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

Emilio P. Mancini◦, Hardik Soni◦, Thierry Turletti◦, Walid Dabbous◦, Hajime Tazaki*

◦INRIA, France, *University of Tokyo, Japan

DCE Objectives

DCE aims to increase the realism of experimentation results of traditional simulators by running real existing protocol implementations inside a network simulator.

Using DCE, a user can

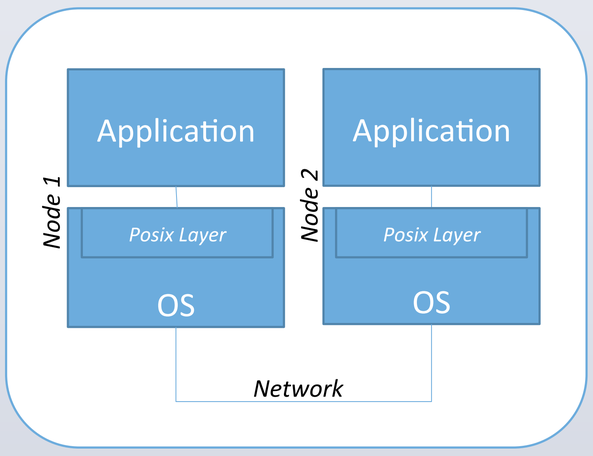
- Avoid re-implementing possibly complex protocols for simulation or use the same code installed in production environments (e.g., DCTCP);
- Obtain reproducible and realistic experimentation results;
- Debug and test code easily within a controlled environment.

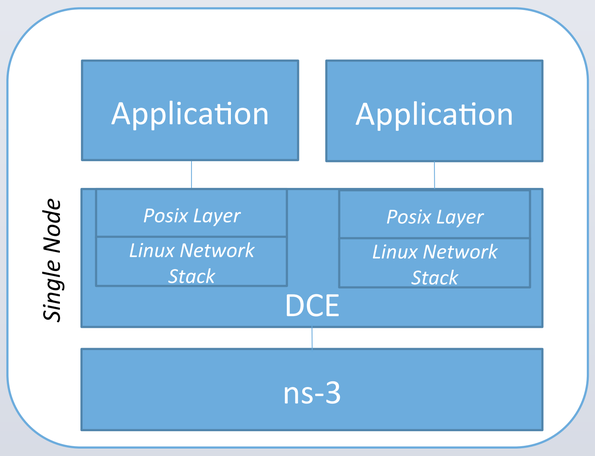
Supported kernels and applications:

- FreeBSD, Linux (several versions from 2.6.36)
  - IPv4, IPv6, TCP/UDP, DCCP, mptcp, Mobile IPv6, etc.
- Quagga, PARC ccnx, iperf, ip, ping/ping6, umip, bind9, unbound, thttpd, bittorrent.

How It Works?

- DCE dynamically links the applications with a re-implemented subset of POSIX APIs interfacing the ns-3 simulator.
- Applications are started in a sand-box environment, which make them controllable and easier to debug: every simulated node has a private file system and environment.
- DCE simulation scripts (in C++ or Python) define the experimentation scenario including the network environment.
- DCE can use the ns-3, Linux or FreeBSD kernel network stacks.





Demonstration 1: Kernel protocols and scalability

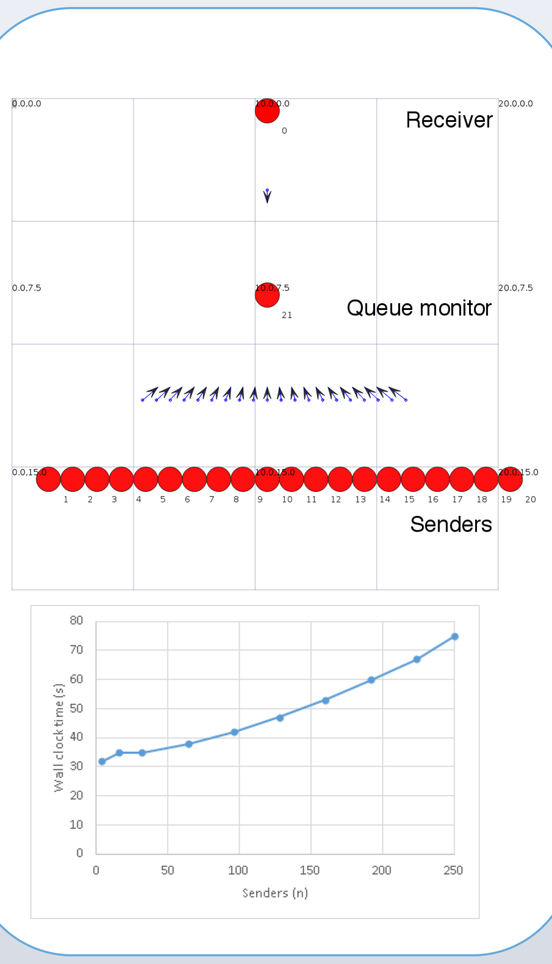
In this demonstration we show how to study a custom protocol in DCE. We run the experiments with DCTCP, an enhancement of TCP for data centers.

DCE allows full reproducible tests on the production code in a number of simulated scenarios.

We use a patched Linux Kernel, the *iperf* utility, and ns-3 tools to analyze the simulation.

Demonstration prerequisites:

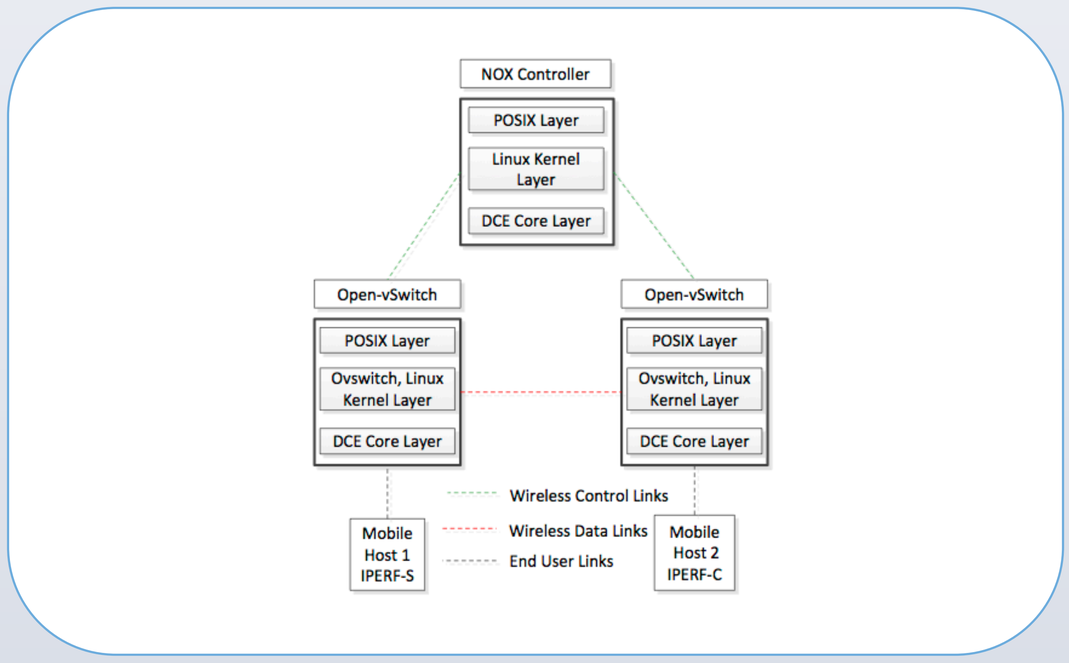
- Update the Linux Kernel code with DCTCP patch:
  1. Checkout the Kernel source code from DCE repository
  2. Apply the patch to the code
  3. Recompile the Kernel using the special *sim* architecture
  4. Reference the kernel binary in the simulation script
- Write a ns-3 network scenario deploying client/ server application binaries on the nodes.



Demonstration 2: Software Defined Wireless Networks

Today there is no solution to evaluate and debug in a realistic way SDN network systems on wireless environments.

As SDN systems are distributed on several nodes, it is hard to debug them in real testbeds, which cannot be fully controllable. Combined with the large set of wireless protocols offered by ns-3, DCE provides an evaluation framework that allows reproducibility of experimentation results, easy debugging and higher scalability than Container Based emulation frameworks. Full support for SDN is expected to be included in the next DCE version.



For further information

- Project Web page: http://www.nsnam.org/overview/projects/direct-code-execution/
- http://yans.pl.sophia.inria.fr/trac/DCE/wiki/dce-dctcp

References

• Hajime Tazaki, Fréderic Urbani, Emilio Mancini, Mathieu Lacage, Daniel Câmara, Thierry Turletti, and Walid Dabbous, **Direct Code Execution: Revisiting Library OS Architecture for Reproducible Network Experiments**, ACM CoNEXT 2013, December 2013.

• Hajime Tazaki, Frederic Urbani, and Thierry Turletti, **DCE Cradle: Simulate Network Protocols with Real Stack for Better Realism**, Workshop on ns3 (WNS3), March 2013.