CommonCode: A Code Reuse Platform for Co-Simulation and Experimentation

Junhee Lee†, Jinsung Lee†, Kyunghan Lee‡, and Song Chong†
† KAIST, Korea {junhee, ljs}@netsys.kaist.ac.kr, songchong@kaist.edu
‡ NCSU, Raleigh, NC, USA, klee8@ncsu.edu

ABSTRACT

Experimentation of a wireless network protocol over the air is of significant interest. However it is rarely performed compared to the simulation because of the difficulties in coding and debugging as well as scalability and repeatability. CommonCode is a unique protocol validation platform reusing the simulation codes for the experiments. It helps fast, convenient and accurate validation of protocols.

General Terms

Platform, Performance

Keywords

802.11, Cross-Layer, Overlay, Simulation, Experiment

1. INTRODUCTION

How many wireless network protocols are experimented and simulated? It might be hard to say exact numbers for the question, but it is clear that most of the protocols developed in research community have been validated by simulation, and only few of them are done by implementation and experiment. The main reason is two-fold: the simulation is 1) rapid to implement and 2) easy to verify the protocols. Especially, the convenience in coding and debugging as well as in execution with scalability and repeatability has attracted the researchers. As a result, NS-2[1], OPNET[2], GloMoSim[3], and other simulators are available at the moment.

However, some wireless communication research areas involving physical layer(PHY) techniques or network layer protocols interacting with PHY require real-world experiments since the PHY model given to the simulator is incomplete to capture the entire reality. For that reason, PHY experimentation platforms such as GNU Radio[4] and WARP[5] are getting more popular.

Nonetheless, are we satisfied with the two distinct validation platforms? So far, maybe yes but we need more. It is known that strict separation of network layers fails to provide the best performance. For example, a cross-layer based resource allocation (e.g., max-weight [2] and references therein) is known to be throughput-optimal, which essentially requires information exchanges across MAC/Routing/Transport layers. This, in turn, demands an *all-in-one validation* platform connecting all layers in a way as easy as simulators and as realistic as PHY experiment platforms. To this end, we propose a novel experimentation platform *CommonCode*, a code-reuse platform for simulations and experimentations.

The CommonCode (CC, in short) uniquely allows the researchers to reuse simulation codes used in GlomoSim for the experiment with multiple Linux-based wireless nodes *without any code changes*. Thereby, the complicated efforts involving system programming in MAC/Routing/Transport layers to re-build the protocols on the hardware become unnecessary. As a demonstration, we have successfully built an experimental testbed [1] for evaluating the optimal resource allocation protocol in a very short period. We believe that CC can expedite the realistic validation of novel protocols, especially cross-layer ones.

2. COMMONCODE IMPLEMENTATION

CC reuses the simulation code for the experimentation through the dual architecture as shown in Fig. 1. In CC, the *CommonCode Base* is operated with the software and the hardware adaptors for simulations and experiments. Three major components of CC: software adaptor, hardware adaptor, and event scheduler are described in detail.

Hardware Adaptor: This component is designed for interfacing CC Base and the device driver. It not only conducts packet conversion, but also controls some hardware functions through an user-space programming. More specifically, it converts a packet generated in CC Base into an actual one destined to a physical interface by

[1]http://www.isi.edu/nsnam/ns
[2]http://www.opnet.com
[3]http://pcl.cs.ucla.edu/projects/glomosim

[4]http://gnuradio.org
[5]http://warp.rice.edu/news.php

ACM CoNEXT Student Workshop, November 30, Philadelphia, USA.

Table 1: Comparison of the Existing Protocol Validation Platforms

	CommonCode	NS[6]	OpenWRT	WARP	GNU Radio
Simulation Code Reusability	Yes	Yes	No	No	No
Network Protocol Library	Abundant	Abundant	Abundant	Limited	Limited
MAC Layer Programmability	Yes	No	No	Yes	Yes
PHY Layer Programmability	No[7]	No	No	Yes	Yes
Program Language	C	C++, OTcl	C	C, Verilog, VHDL	Python, C++
Hardware Platform	General Machine	General Machine	General Machine	Dedicated Hardware	High Performance Machine

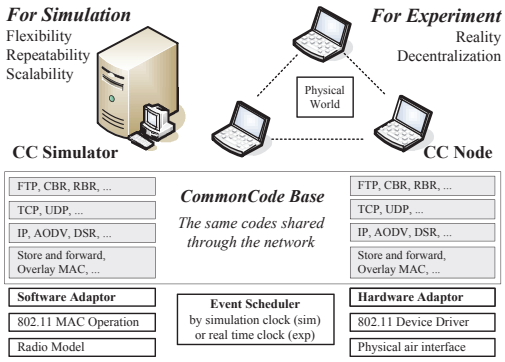


Figure 1: CommonCode Architecture

encapsulating with a real packet format. In addition, it controls queueing and transmission of actual packets more easily through a user-space level MAC implementation (i.e., Overlay MAC) by nullifying kernel level queueing (via programming with raw socket libraries).

Software Adaptor: This is designed for interfacing CC Base and low-level simulation components of 802.11 MAC as well as radio model. It is implemented to imitate all of device driver control functions of the hardware adaptor for the code reuse. For example, it encapsulates the packets with a simulation packet format.

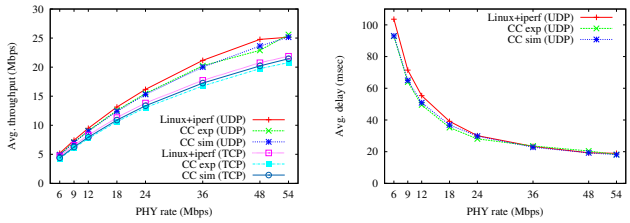
Event Scheduler: This mainly converts the simulation clock to the real system clock for experiments. There is a possibility of having some delay in the real system clock when an event requires heavy computation impossible to be handled within a given inter-event duration. However, it rarely happens and is not critical in general cases of considering 54 Mbps links (802.11a) and distributed protocols.

The comparison of the features between CC and the existing experimental validation platforms is given in Table 1.

3. PERFORMANCE EVALUATION

We verify validity and capability of CC by providing experimental results comparing the performance of iperf on Linux (Linux+iperf), CC simulation, and CC experiment. For real experiments, we use commercial netbook platform (1.66GHz Atom CPU and 1GB RAM), equipped with two 802.11 mini PCI-Express NICs (Atheros chipset). We set up the same 802.11 control parameters for CC simulations and experiments. We test a (saturated) single-hop flow using two nodes in which the same programming code is separately installed. We compare CC simulations and CC experiments with real experiments in pure Linux settings, "Linux+iperf".

Fig. 2 shows two major performance metrics, throughput and packet delay for saturated UDP and TCP traffic for various PHY rates. The graph verifies the successful code-reuse of CC for the experiment. The performance gap between CC simulation and CC experiment is negligible, meaning that the computational overhead (e.g., packet conversion and clock conversion) in CC experiment is essentially minimized.



(a) Throughput (b) Delay

Figure 2: Performance comparison for saturated traffics (UDP and TCP) with different 802.11a PHY rates

4. CONCLUSION AND FUTURE WORK

We implemented a novel co-simulation and experimentation platform, CommonCode, which takes both advantages of existing simulators and experimentation platforms at a time. Our future work includes an application adaptor which injects real user traffic into CC-Base to make a node with CC work as a router. We further consider a unification with PHY experimentation platforms.

5. ACKNOWLEDGEMENT

This research was supported by the MKE, Korea, under the ITRC support program supervised by the NIPA (NIPA-2010-(C1090-1011-0004)).

6. REFERENCES

[1] J. Lee et al. Implementation of optimal csma. Technical report, KAIST. http://netsys.kaist.ac.kr/main/publications/Resources/implement.pdf.

[2] M. J. Neely et al. Fairness and optimal stochastic control for heterogeneous networks. In *IEEE Infocom*, 2005.

[6]NS-2 also supports TCP implementation in Linux, http://netlab.caltech.edu/projects/ns2tcplinux/ns2linux/index.html

[7]Possibly, the core of CC can be extended to connect simulation codes to state-of-the-art PHY experiment platforms, e.g., WARP or GNU Radio.