

Wireless link emulation in OneLab

Extended Abstract

M. Carbone, G. Cecchetti, L. Rizzo

Dip. Ingegneria dell'informazione
Università di Pisa

Email: {marta.carbone,rizzo}@iet.unipi.it, g.cecchetti@sssup.it

F. Checconi, A. L. Ruscelli

Retis Lab.

Scuola Superiore S. Anna

Email: {f.checconi,a.ruscelli}@sssup.it

I. INTRODUCTION

This paper presents a work in progress to add emulation of IEEE 802.11 [1], [2] wireless links to the OneLab¹ system.

The goal is to extend OneLab with an emulation component (an external box sitting on the link between a node and the rest of the network) which can reproduce the effects of a wireless links on live traffic. In this way one can run repeatable experiments under controlled conditions of the wireless network. Key goals of the project are ease of use and configuration, and minimal impact on existing code, so that experiments can be re-run without the need to recompile source code.

Modifications to the existing OneLab architecture [3] include the addition of new entities, namely the emulation boxes, and corresponding changes to the PLC² management interface to link nodes and emulation boxes. Additionally, we have defined a simple yet flexible API that can be used at various level (configuration scripts or running experiments) to modify the features of the emulated link. Different experiments can run concurrently with different [emulated] network conditions, if they need to.

In the following, we describe the changes to the OneLab architecture, and the API used to configure the emulated link. Finally, Section III discusses the methodology used to model the wireless link and estimate the configuration parameters for the emulator.

II. PLC EXTENSIONS

The addition of the emulation box requires instructing the PLC software to store information about the new component, and its connection to the nodes. Suitable fields have been added to the database tables on the central site, and the management software has been updated to handle this information. The PLC software is also in charge of producing software images for the emulation box, same as it is done for other OneLab nodes. Configuration and authentication information is also distributed to the emulation boxes using the existing mechanism and APIs.

The emulation box is implemented by a modified version of the dummynet [4] software, which acts as a network bridge able to shape and delay the traffic flowing through it.

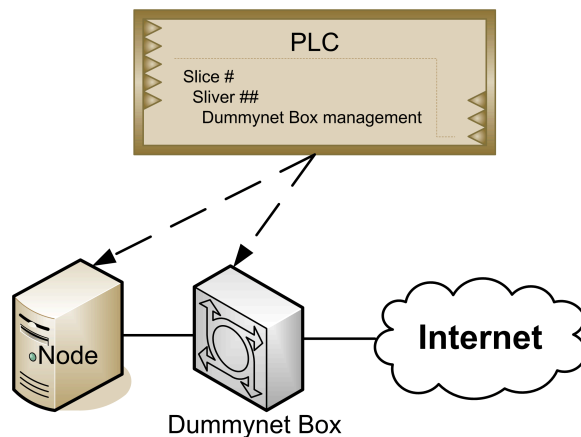


Figure 1. Dummynet Box and OneLab network

This software is installed on a single board PC, sitting on the link between a OneLab node and the rest of the network. This system is called “*Dummynet box*”, and can be dynamically configured to act selectively on the traffic flowing through it. As a consequence, control traffic to manage OneLab nodes will flow unaffected, whereas traffic for each sliver³ can be shaped to emulate a single or multiple independent wireless networks.

Running an experiment requires the configuration of the operating conditions of the wireless network. The simplest way to achieve this is to invoke, right before the start of the experiment, a one-line command such as:

```
emu802.sh cli 5 bw 1Mbps sn 38dB port 5210
```

This will in turn call the appropriate programs to configure the local interface and talk over a secure channel to the dummynet box to set up the emulation parameters for this sliver. In this way, users have flexibility in setting up the operating environment without the need to communicate with the central site, and without the need to recompile the experiments' code. Experiments in need of more sophisticated control, e.g., time-varying network conditions, can achieve it by invoking a configuration command during the experiment itself.

¹OneLab EU project - Contract n. 034819

²A PLC is the central management software of a PlanetLab/OneLab network.

³A sliver is a set of resources allocated on a single Planetlab node to a “slice”.

III. A MODEL FOR WIRELESS DUMMYNET

The accuracy of the results of an experiment depends of course on the behavior of the emulation box. Here there are conflicting requirements between the accuracy and overhead of the emulation. Modeling all the features of the MAC layer is outside the scope of this work and of the platform itself, because there are already artifacts (such as process scheduling, other activities of the operating system, uncertainty on the behavior of the rest of the network) that act on a coarser timescale. For the same reasons, we do not want to model individual events on a microsecond timescale, but only look at their aggregate effects. Thus, our focus is on effects of interest for an experiment running on OneLab, such as:

- the number of active clients. This influences the medium access delay, and also the bandwidth available for each client. Also, some system may exhibit trashing effects with a large number of clients, e.g., due to too few slots in the associations' list on the access point;
- the S/N ratio. This is generally used by the card's hardware to drive the rate adaptation mechanism (hence has a direct impact on the available bandwidth). Also, low S/N ratios cause an increase in packet losses.

The effect of such parameters will be evaluated in part by analyzing the relevant MAC protocol standards, and in part by looking at wireless traces [5]. The result of the evaluation will be used to map the (high level) parameters of `emu802.sh` into the (low level) configuration arguments for the dummynet box described below.

A. Protocol analysis

From the study of the 802.11 protocol we can infer an initial set of high level parameters that affect individual packet transmissions:

`PROTOCOL_OVERHEAD` (μs)

is the time while the channel is not used for actual data transmission due to arbitration intervals, preambles, and mandatory protocol fields (ACKs, typically). Irrespective of the data rate, each transmission event incurs an additional delay ranging between 100 and 600 μs , depending on some protocol parameters;

`DATA_RATE` (*bit/s*)

is the raw bit rate used for payload transmission, corresponding to one of the supported 802.11 rates as configured by the user and the rate adaptation algorithm (of course, the achievable data rate is always lower because of the protocol overhead, contentions and retransmissions).

B. Evaluation of Wireless parameters for Dummynet

Additionally, we have analyzed some existing MAC-layer traces of IEEE 802.11 protocol, trying to evaluate the impact of phenomena such as number of clients and S/N ratio on parameters such as contention delays, error rates, and actual data rates. Initially we plan to model these effects with two additional parameters, namely:

`CONTENTION_RATE` (*probability, 0..1*)

is the probability that acquiring the channel fails because of contention, causing the station to wait until the winner of the arbitration completes its transmission. This depends on the number of active clients, and influences the transmission delay;

`TX_FAILURE_RATE` (*probability, 0..1*)

is the probability that a packet transmission fails (meaning that the sender does not receive the 802.11 ACK) and so a retransmission is required. Failures of this kind are generally due to poor s/n ratios. Traces report failures (requiring retransmissions) as high as 20%.

Dummynet emulator already implements `DATA_RATE` parameter, but it is necessary to extend the kernel side with appropriate procedures to account for the (deterministic) extra time needed to complete a packet transmission (due to `PROTOCOL_OVERHEAD`) and to add the (probabilistic) delays due to non-zero `CONTENTION_RATE` and `TX_FAILURE_RATE`.

To evaluate the delay due to `CONTENTION_RATE` the trace analysis has highlighted the influence of active nodes number figuring out how many nodes were associated to a single access point, dividing the associated ones from the unassociated ones [6].

To compute the delay due to `TX_FAILURE_RATE` we measured the delays between data transmissions and their corresponding ACKs both in case of successful transmission and in case of re-transmission. While we can measure the difference between the instant of received ACK and the one of data transmission we can only estimate the time awaited by the wireless station before sending data. The distribution of this waiting time has been obtained by evaluating retransmission times and silence period. The former is particular useful because a retransmission procedure shows the time elapsed between two consecutive data packets due to a missing ACK. If we subtract the ACK-timeout we obtain the station waiting time.

The full version of this paper will present the analytical data collected with the relationship with depicted parameters.

REFERENCES

- [1] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 1997.
- [2] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification: Higher-Speed Physical Layer Extension in the 2.4 GHz Band*, IEEE Std. 802.11b, 1999.
- [3] https://www.planetlab.org/doc/plc_api.
- [4] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM Computer Communication Review*, January 1997.
- [5] <http://crawdad.cs.dartmouth.edu>.
- [6] M. Rodrig, C. Reis, R. Mahajan, D. Wetherall, and J. Zahorjan, "Measurement-based characterization of 802.11 in a hotspot setting," in *E-WIND '05: Proceeding of the 2005 ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis*. New York, NY, USA: ACM Press, 2005, pp. 5–10.