Demo: Simulating a 6TiSCH Network using Connectivity Traces from Testbeds

Yasuyuki Tanaka, Keoma Brun-Laguna, Thomas Watteyne

▶ **To cite this version:**

Yasuyuki Tanaka, Keoma Brun-Laguna, Thomas Watteyne. Demo: Simulating a 6TiSCH Network using Connectivity Traces from Testbeds. CNERT - IEEE INFOCOM Workshop on Computer and Networking Experimental Research using Testbeds, Apr 2019, Paris, France. hal-02266552

HAL Id: hal-02266552
https://inria.hal.science/hal-02266552

Submitted on 14 Aug 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Demo: Simulating a 6TiSCH Network using Connectivity Traces from Testbeds

Yasuyuki Tanaka, Keoma Brun-Laguna, Thomas Watteyne
Inria, Paris, France.
{yasuyuki.tanaka,keoma.brun,thomas.watteyne}@inria.fr

***Abstract*—The 6TiSCH simulator is an existing Python-based simulation tool that captures the full behavior of 6TiSCH, the Industrial IoT protocol stack standardized by the IETF. The existing 6TiSCH simulator uses a radio propagation model. In this demo, we present an extension to the 6TiSCH simulator which allows a simulation to be run against connectivity traces previously gathered on testbeds and real-world deployments. We demonstrate four elements. First, Mercator, the OpenWSN-based tool we developed to collect connectivity traces from different testbeds. Second, K7, the generic format we defined for these connectivity traces. Third, the set of 17 connectivity traces we gathered from testbeds and real-world deployments, and which are publicly available. Fourth, the extension of the 6TiSCH simulator which enables it to replay K7 connectivity traces rather than using a propagation model. Using connectivity traces for simulation is a way to increase the confidence the result are representative of a real-world deployment. Furthermore, it allows better repeatability than re-running an experiment on a testbed where the connectivity necessarily changes over time.**

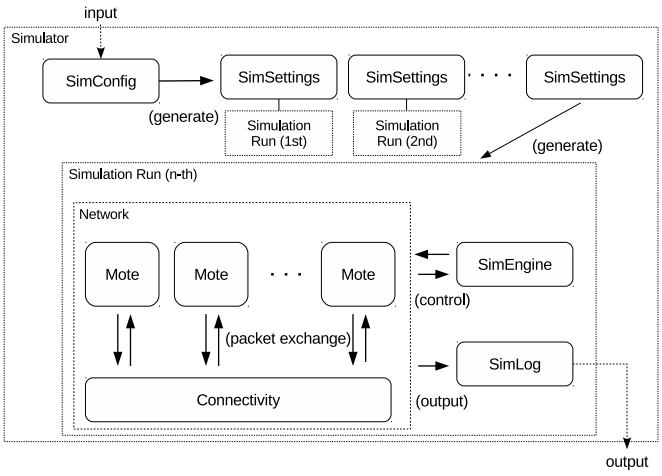

Fig. 1: The architecture of the 6TiSCH Simulator.

I. Introduction

Time Synchronized Channel Hopping (TSCH) is a Medium Access Control (MAC) mode of IEEE802.15.4 [1] to address industrial use cases. The core part of TSCH is a combination of time-division multiple access and channel hopping. On top of TSCH, the IETF 6TiSCH working group [2] is standardizing a complete IPv6 protocol stack, which includes routing, security and scheduling protocols.

Network characteristics of a 6TiSCH network depends heavily on its communication schedule. The schedule defines which mote can perform transmission to whom and when. In addition to timings, it defines the radio frequency to be used for that communication. If the schedule is collision-free, the network can achieve high reliability. The schedule can also be optimized further for end-to-end latency or battery life. However, the communication schedule cannot be static because the radio environment varies over time and motes may be added or removed. Dynamic scheduling in 6TiSCH networks receives significant attention from the acedemic community.

The 6TiSCH Simulator [3] is developed to accelerate research and development of 6TiSCH related technology, including scheduling algorithms[1]. The simulator implements the full behavior of the 6TiSCH protocol stack. The simulator allows a user to more easily add a new scheduling algorithm, compared to implementing the same on real hardware. On a regular computer, simulations run 10-1000 times faster than running experiments on a testbed. Furthermore, the 6TiSCH simulator is architected to also be run on computer clusters.

One key drawback of simulation-based studies for wireless networking is the need to use a radio propagation model. Such models attempt to represent the radio channel, but necessarily cannot capture a subtle phenomena such as variations of multi-path fading during elements moving, or variations of interference. The danger is that the performance of a protocol stack on a simulator is different than that in a real-world deployment. Instead, we propose trace-based simulation: replay previously collected connectivity traces as a replacement for a radio propagation model.

This demo shows trace-based simulation on the 6TiSCH simulator, including *(1)* Mercator, the tool to collect the traces, *(2)* K7, the generic forma to represent the traces, *(3)* 17 collected traces which are publicly available and *(4)* an extension to the 6TiSCH simulator to run connectivity traces.

II. The 6TiSCH Simulator

The 6TiSCH Simulator is a discrete-event simulator written in Python, which implement the full 6TiSCH protocol stack, depicted in Table I.

Fig. 1 shows the software architecture of the 6TiSCH Simulator. The input to the simulator is a configuration file

[1] As an online addition to this paper, the 6TiSCH Simulator is available at https://bitbucket.org/6tisch/simulator/src under an open-source BSD license

Protocol	Specification
Routing	RFC6550, RFC6552, RFC6206
Fragmentation	RFC6282, RFC4944, draft-ietf-6lo-minimal-fragment-00
TSCH Scheduling	RFC8180, RFC8480, draft-ietf-6tisch-msf-01
Join Process	draft-ietf-6tisch-minimal-security-09
MAC	IEEE802.15.4-2015

TABLE I: The 6TiSCH protocol stack, implemented by the 6TiSCH simulator.

which contains settings such as the number of motes in a network and the scheduling function to be used.

The simulation run is controlled by the discrete-event engine called `SimEngine` and its output data is collected through `SimLog` into a log file. In the network, a frame sent by a mote is propagated to its neighbors through a `Connectivity` instance based on link quality. The link quality (PDR and RSSI) of a given pair of motes are determined by the `Connectivity` class in use. For instance, the `ConnectivityRandom` implements "Pister-hack" propagation model[2].

We extend the 6TiSCH simulator by defining a new subclass to `Connectivity` to replay connectivity traces.

III. Connectivity Trace

In order to fill the gap between simulation and experimentation, we extend the 6TiSCH Simulator with trace-based simulation capability so that the simulator can run with a connectivity trace instead of a propagation model.

The connectivity trace is a time-series dataset of link PDR and RSSI measurements of all possible links. Since it is a time-series, the trace captures the dynamics of radio environment over time. We collect the trace on all available radio frequencies as multi-path fading and external interference impact link quality differently on different frequencies.

We develop a tool called Mercator[3] which enables to collect connectivity traces on a testbed in an automated manner. Mercator configures one mote to transmit while all the other are listening. It has the transmitter send a burst of frames; the receivers report the frame counter and RSSI of each frame received. After the burst, the PDR (portion of frames received) and the mean RSSI values are computed for each receiver, indicating the quality of the link between the transceiver and that receiver. This process repeats for each channel, and by having all motes be transmitter, in a round-robin fashion. This is automated by Mercator, so connectivity traces can be gathered for days at a time without requiring human intervention.

Connectivity traces are formatted according to K7, a CSV-based format for multi-channel radio connectivity traces[4].

We extend the 6TiSCH simulator with `ConnectivityK7`, a new `Connectivity` sub-class, which replays a K7 connnectivity trace as a replacement for a propagation model. This enables the user to perform experiments efficiently by simulation, knowing that the results represent a real-world deployment.

[2] https://people.eecs.berkeley.edu/~pister/290Q/09sp/09sp%20hw3.htm

[3] As an online addition to this paper, Mercator is available at https://github.com/openwsn-berkeley/mercator under an open-source BSD.

[4] https://github.com/keomabrun/k7

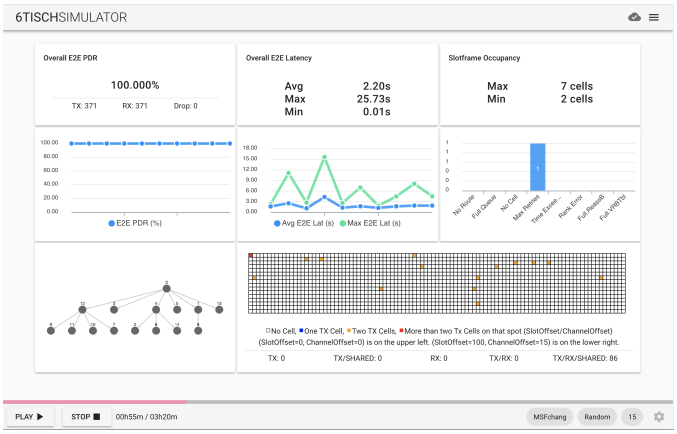

Fig. 2: Screenshot of the web interface of the 6TiSCH Simulator. A user can see performance indicators as the simulation is running, including end-to-end PDR (Packet Delivery Ratio), end-to-end latency, network topology, and the TSCH schedule.

IV. Demonstration

We demonstrate the different elements described above, using a pair of computer screens.

On one computer screen, we show Mercator running live on the FIT IoT-lab Grenoble testbed, and the connectivity trace that is being recorded. We place a QR code which points to the repository of 17 connectivity traces which have been gathere on 3 testbeds and 3 real-world deployments, and which are available[5].

One the second screen, we demonstrate the web interface of the 6TiSCH simulator as it is running. The simulation we run is replaying a connectivity trace previously recorded on the FIT IoT-lab Strasbourg testbed.

V. Conclusion

This demo shows an extension to the 6TiSCH simulator which allows it to replay previously-recorded connectivity traces, as a replacement for a propagation model. We believe trace-based simulations are an elegant way of creating simulation results that represent the real-world while maintaining perfect repeatability.

References

[1] *802.15.4-2015 - IEEE Standard for Low-Rate Wireless Networks*, IEEE Std., April 2016.

[2] T. Watteyne, M. R. Palattella, and L. Grieco, *Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement*, IETF Std. RFC7554, May 2015.

[3] E. Municio, G. Daneels, M. Vucinic, S. Latre, J. Famaey, Y. Tanaka, K. Brun, K. Muraoka, X. Vilajosana, and T. Watteyne, "Simulating 6TiSCH Networks," *Wiley Transactions on Emerging Telecommunications (ETT)*, 2018.

[5] https://github.com/keomabrun/dense_connectivity_datasets