Trace-Based Simulation for 6TiSCH

Yasuyuki Tanaka, Keoma Brun-Laguna, Thomas Watteyne

Received <day> <Month>, <year>; | Revised <day> <Month>, <year>; | Accepted <day> <Month>, <year>

DOI: xxx/xxxx

SPECIAL ISSUE ARTICLE

Trace-Based Simulation for 6TiSCH

Yasuyuki Tanaka*[1,2] | Keoma Brun-Laguna[1] | Thomas Watteyne[1]

[1]EVA, Inria, Paris, France
[2]Corporate Research and Development Center, Toshiba, Kanagawa, Japan

Correspondence
*Email: yasuyuki.tanaka@inria.fr

Present Address
Inria, 2 Rue Simone Iff, 75012 Paris, France

Abstract

Simulation is a powerful tool for development of wireless networks. One key drawback of simulation-based studies for wireless networking is the need to use a radio propagation model. In this paper, we propose trace-based simulation for 6TiSCH (IPv6 over the TSCH mode of IEEE802.15.4e). **We make the 6TiSCH Simulator run against connectivity traces collected in a real-world deployment.** Our evaluation shows the trace-based simulation can yield almost the same results as experiments, with 1,200 times faster speed in execution time.

KEYWORDS:
IEEE802.15.4; TSCH; 6TiSCH; simulator; trace-based simulation

1 | INTRODUCTION

Simulation is a powerful tool to accelerate research and development of networks, especially for wireless sensor networks where constrained devices are involved. Simulation enables developers to focus only on essential parts of protocol behaviors without being bothered by implementation details of constrained devices. In general, simulations run way faster than running experiments. These properties best fit an early development stage.

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. **Although physical layer phenomena in wireless communications can be accurately modelled using statistics, it is nearly impossible to simulate subtle variations of the phenomena such as multipath fading.** 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.

In this paper, we implement a set of software for trace-based simulation of 6TiSCH[1] networks. 6TiSCH realizes wireless multihop IPv6 networks over the TSCH (Time Slotted Channel Hopping) mode of IEEE802.15.4. We select the 6TiSCH Simulator as a simulator to extend since ns-3[1] does not support 6TiSCH, and Cooja[2] and OpenSim[3] are tightly coupled with specific 6TiSCH stacks, Contiki and OpenWSN. The contribution of this paper is three-fold: *(1)* a framework of software suite for trace-based simulation, *(2)* enhancements to the 6TiSCH Simulator in terms of supported protocols, and adding a trace-based simulation feature, *(3)* OpenTestbed[4]-support to Mercator[5].

The remainder of this article is organized as follows. Section 3 describes the proposed software suite for trace-based simulation. Section 4 shows evaluation results comparing trace-based simulations and experiments in a testbed. Section 5 analyzes the results. Section 2 introduces related work. Finally, Section 6 concludes this paper.

[1]https://www.nsnam.org/
[2]https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja
[3]https://openwsn.atlassian.net/wiki/spaces/OW/pages/13434892/OpenSim
[4]https://github.com/openwsn-berkeley/opentestbed
[5]https://github.com/openwsn-berkeley/mercator

2 | RELATED WORK

Noble *et al.*[2] does one of earliest and most inspiring works. The authors propose a technique called *trace modulation* that reproduces observed end-to-end characteristics of WaveLAN. They build a network model based on trace information collected in a real WaveLAN network. Packet delay and drop in WaveLAN are *emulated* over an Ethernet link by the model implemented in a pseudo network device of NetBSD. This approach suits best a system where a wireless link is provided to a device as part of infrastructure. In other words, network performance properties, for instance packet drop rate, are given in such a network. This is not the case in a multi-hop wireless network where behaviors of devices under evaluation determine network performance.

Trace-based simulation is used in many research works on wireless sensor networks. Marchiori *et al.*[3] evaluate CTP (Collected Tree Protocol) by trace-based simulation with their own simulator called WsnSimPy. Jacobson *et al.*[4] implement a trace-based simulation model on ns-2, and evaluated flooding protocols. Lamera *et al.*[5] propose a trace-based simulation on ns-3, evaluating it with WiFi. These papers compare their simulation results with experiments conducted in the testbeds. Although they demonstrate how accurate trace-based simulation compares to pure simulation, the topology used in each experiment is too simple or not realistic: a linear topology of three nodes, a grid topology of 50 nodes without any obstacle, and a point-to-point link of a pair of nodes. Instead, we evaluate our trace-based simulation implementation against traces collected in an office building, with which a multi-hop network is formed.

Trace is also used in troubleshooting in a wireless multihop network. Qui *et al.*[6] propose a troubleshooting framework using trace-based simulation. Unlike the trace-based simulation discussed in Section 3, their framework collects traces from a network where applications are running. Trace-based simulation is used to produce an expected network performance based on the traces, which will be compared with observed network performance to detect faults and to identify their root causes. Applying this framework to 6TiSCH is interesting and one of our possible future work.

Regarding trace collecting tools, SCALE[7] and Connectivity Daemon[8] collect traces in a similar manner to Mercator. However, they are not as extensible to other implementations or testbeds as Mercator. Since SCALE is built in the EmStar programming model, devices need to have a special interface, *MoteNic* or *Udpd*, to intereact with SCALE. Connectivity Daemon runs only in MoteLab as it is designed for. Unlike them, Mercator requires devices only to handle simple serial commands. Mercator supports FIT/IoT-LAB and OpenTestbed, which are totally different infrastructures.

3 | A SOFTWARE SUITE FOR TRACE-BASED SIMULATION

Easiness in collecting connectivity traces is as important as capability of trace-based simulation in a network simulator. In this section, we introduce a software suite for trace-based simulation, which adopts the 6TiSCH Simulator as a network simulator. The tools in Section 3.1 work directly for use-cases other than 6TiSCH. The way of implementing trace-based simulation explained in Section 3.2 can be carried over to other network simulators.

3.1 | Tools for Connectivity Trace Collection

OpenTestbed[9] is an open-source platform allowing developers to build their own testbed at any place with off-the-shelf components such as a Raspberry Pi single-board computer and OpenMote B[6] low-power wireless devices.

Figure 1 shows an OpenTestbed node called OTBox. An OTBox consists of one Raspberry Pi and four OpenMote B boards. The Raspberry Pi controls the four boards via serial connections, and connects to a management network using 5GHz WiFi. The key part of OpenTestbed is a Python program called `otbox.py` running on the Raspberry Pi, which provides APIs to control the OpenMote B boards over MQTT. Thanks to the APIs, a developer can remotely flash a firmware, and access the serial connections of the wireless devices.

Mercator[10] is an automated tool designed to collect connectivity traces in a deployment site. We make Mercator able to run over OpenTestbed, as well as over FIT/IoT-LAB[7] testbeds. Mercator consists of firmware and software, which are publicly available under the 3-Clause BSD license. The firmware is flashed to all the devices in the deployment. Then, the software running on a PC orchestrates measurements of radio link quality communicating with the devices via serial connections. However, the PC does not need to have a physical connection to the devices.

[6] https://www.industrialshields.com/shop/product/is-omb-001-openmote-b-721
[7] https://www.iot-lab.info/

FIGURE 1 OpenTestbed (OTBox)

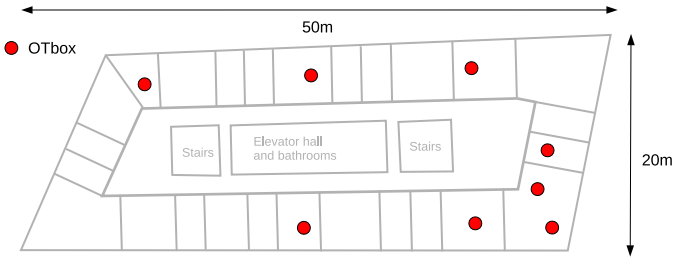

FIGURE 2 Floor map with OTBox positions

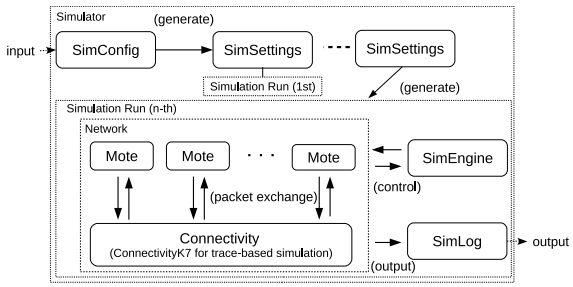

FIGURE 3 Architecture of the 6TiSCH Simulator

Protocol	Specification
Routing	RFC6550, RFC6552, RFC6206
Fragmentation	RFC6282, RFC4944, draft-ietf-6lo-minimal-fragment
TSCH Scheduling	RFC8180, RFC8480 draft-ietf-6tisch-msf
Join Process	draft-ietf-6tisch-minimal-security

TABLE 1 Protocols implemented in the 6TiSCH Simulator

Mercator selects one of the devices as the transmitter, and makes the rest become receivers. The transmitter broadcasts a specified number of frames with a specified transmission power, at a specified radio channel. Every time receiving a frame sent by the transmitter, a receiver reports RSSI to the software. At the end of the set of transmissions, the software **has** data to compute PDR and a mean RSSI of all links from the transmitter to the receivers. This process continues changing the transmitter and the channel in a round-robin fashion.

3.2 | 6TiSCH Simulator with Trace-Based Simulation

The 6TiSCH Simulator[11] is an open-source simulator written in Python. **Figure** 3 shows the architecture of the 6TiSCH Simulator. The input to the simulator is a configuration file which contains settings such as the number of motes in a network and the scheduling function to be used. In order to increase accuracy of simulation results, we enhance the 6TiSCH Simulator improving its supported protocols and adding the trace-based simulation capability.

Table 1 shows supported protocols of the latest 6TiSCH Simulator, which we identified as the minimal set of protocols to simulate behaviors of a 6TiSCH network correctly. We improve existing implementation of the protocols, and added missing protocols. We refine the TSCH layer as well. For instance, we introduce the frame pending bit feature defined in IEEE802.15.4.

The `Connectivity` class simulates radio propagation based on link quality managed by the simulator. Link quality is defined as a pair of PDR (Packet Delivery Ratio) and RSSI (Received Signal Strength Indicator), which is set to a directional link of each pair of nodes. By default, link quality values are calculated at the beginning of a simulation using a "Pister-hack" propagation model[8] based on the positions of motes. In a trace-based simulation, a new Connectivity subclass called ConnectivityK7 loads link quality values from a given trace file formatted in K7 during the initialization. The trace file is a time-series dataset of link PDR and average RSSI measurements of all possible links distinguished by a tuple of a source MAC address, a destination MAC address, and a radio channel. Mercator provides a converter from raw connectivity traces into K7 format. With `ConnectivityK7`, the radio environment is perfectly simulated as observed while collecting the trace in a real testbed or deployment.

When a device transmits a frame and there is no other transmissions, the simulator takes the PDR from the transmitter to each device which listens on the channel at that time, and determines whether the transmission succeeds by comparing the PDR with a uniform random value selected between 0 and 1 including both ends. If there are other transmissions, the receiver is locked onto the first reached transmission, and the others are treated as interfering transmissions. Then, the simulator computes SINR

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

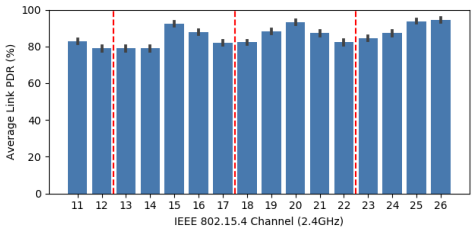

FIGURE 4 Average link PDR of each channel: vertical dotted red lines indicate frequencies of WiFi channels, 1, 6, and 11.

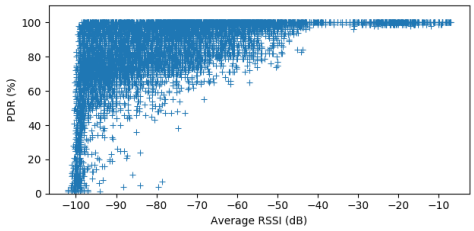

FIGURE 5 Waterfall plot: a scatter plot of measured link PDR as a function of mean RSSI. The mean PDR value with standard deviation is depicted in the orange color.

Parameter	Settings
TSCH EB portion	5, 10, or 20
RPL DIO portion	10
RPL DAO portion	60
Scheduling Function	**draft-ietf-6tisch-msf**
TSCH slotframe length	101 slots
TSCH slot duration	20 ms
TSCH max. retransmission count	15

TABLE 2 Evaluation Settings

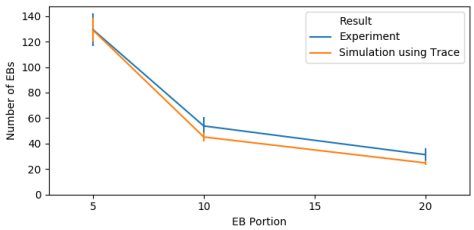

FIGURE 6 # of beacons w/ 95 % confidence interval (CI)

(Signal to Interference plus Noise Ratio) with RSSIs of incoming links from the transmitters, which is converted to a PDR using a predefined PDR-RSSI conversion table. The values in the conversion table are obtained from measurements with an actual radio device. The simulator decides whether the locked-on transmission succeeds or not by comparing the resulting PDR with a random value. There is no difference in this process between the default Connectivity class using the Pister-hack model and ConnectivityK7.

4 | EVALUATION

4.1 | Deployment

We conducted a series of experiments in a testbed built on a floor of an office building, using OpenTestbed. Figure 2 is a floor map indicating the positions of OTBoxes. The floor consists of 26 offices and meeting rooms, organized around a core which hosts elevators, bathrooms and stairs. We deploy eight OTBoxes in offices on the floor. Four OpenMote B boards are connected to each OTBox, the configuration of which is the same as shown in Figure 1 . In total, 32 OpenMote B boards are deployed.

We run Mercator in the testbed to collect traces, for all the available 16 channels for 2.4 GHz of IEEE802.15.4. The measurement is done between 9:38:20 PM and 11:08:15 PM of June 17, 2019, over all the available 16 channels. We generate a *K7* file for the trace-based simulation from the raw data.

We plot the raw data in the manners introduced by Brun-Laguna *et al.* [10]. Figure 4 shows average link PDR of each channel. Link PDR values around WiFi channels are degraded. This reflects perfectly the fact there are WiFi networks deployed in the office building at WiFi channel 1, 6, and 11, which are considered to interfere with IEEE802.15.4 communications. Figure 5 is a scatter plot of PDR, taken from all the frequencies, as a function of RSSI. Since four nodes are deployed in one OTBox in close vicinity, good PDR values with small standard deviations are observed above -40 dBm. There is no strong interference on specific links because there is no large “dips” found in the plot.

4.2 | Evaluation Configuration

We run experiments and simulations with the settings shown in Table 2 . *TSCH EB portion*, *RPL DIO portion*, and *RPL DAO portion* are parameters to determine how frequently a node transmits EB (Enhanced Beacon), DIO (Destination-Oriented

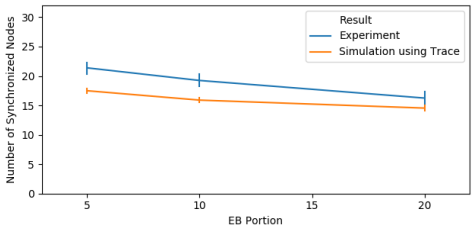

FIGURE 7 # of synchronized nodes w/ 95 % CI

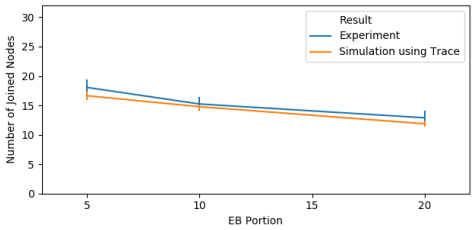

FIGURE 8 # of joined nodes w/ 95 % CI

Directed Acyclic Graph Information Object), and DAO (Destination Advertisement Object) respectively, which are control packets of 6TiSCH[1]. These parameters are OpenWSN-specific.

At the beginning of each experiment or simulation, every node except the root is disconnected and desynchronized from the network. 6 min after the network starts, the experiment or simulation ends. Note that, while the connectivity trace file is taken from the same testbed where experiments are conducted, the quality of radio links during the experiments is never exactly the same as seen in the connectivity trace file. For each simulation run, a different random seed is chosen. This introduces *randomness* to the simulated radio environment even if it is built with the same trace file for each simulation.

For the experiments, we use the development version of OpenWSN[9]. As we mentioned, we use the same set of 32 OpenMote-B boards as for the trace collection. The experiments are grouped by EB portion (5, 10, and 20), and conducted separately during three nights, June 15, 16, 17 of 2019. Key events happened in the OpenWSN firmware, such as beacon transmission, synchronization with the network, and completion of the join process, are recorded through the serial line of a device.

For simulations, we use the version 1.2.1 of the 6TiSCH Simulator with modification to incorporate two OpenWSN-specific mechanisms which impact protocol behaviors: the random number generator (`openrandom`) and the probabilistic broadcasting mechanism. Since `openrandom` takes the MAC address of a node as its random seed, we configure the exact same MAC address to a node in a simulation as its corresponding OpenMote-B board in the testbed. The probabilistic broadcasting mechanism is applied to transmissions of beacons, DIOs, and DAOs. We use five computers in a computing cluster system, each of which has 192 GB RAM and two Intel Xeon E5-2650v4 CPUs. Simulations are run in parallel over 120 CPU-cores, in total.

4.3 | Results

After a node completes the join process, it starts transmitting beacons. The root node starts sending beacons just after it is powered on. Figure 6 shows beacon traffic observed for the first 6 min after the network starts. Beacon traffic changes in accordance with the EB portion value in use, in the same manner between the experiments and the simulations. We have slightly more beacons in the experiments. This implies more nodes have joined the network in the experiments than in the simulations.

When a node receives a beacon, it identifies the time slot structure and gets synchronized with the network. As shown in Figure 7 , 12% – 22% more nodes are synchronized in the experiments regardless of the EB portion value. Figure 8 shows the numbers of nodes which complete the join process, called “joined node”. As seen in Figure 6 , slightly more nodes complete the join process in the experiments than in the simulations.

5 | DISCUSSION

The results shown in Section 4 confirm that network performance yielded by the trace-based simulations shows the same trends as the experiments. While it is also confirmed that the simulation does not give perfectly the same results as the experiment even if using connectivity traces, nevertheless the difference between the two sets of results is small enough to be well accepted.

The simulation gives pessimistic results compared to the experiment. We presume this is because some radio properties such as the capture effect are not implemented in the simulation. Lack of such properties in the simulation affects more communications on a shared medium. In other words, if communications are separated in time-wise and/or frequency-wise, which is often the case in TSCH networks, missing radio properties in the simulator do not count much. This explains why the gap seen in Figure 8

[9]The exact revision is the commit hash of 25e2551 on the official OpenWSN firmware repository, https://github.com/openwsn-berkeley/openwsn-fw/tree/develop

is smaller than in Figure 7 . Messages for the join protocol are exchanged on cells having far less contentions than the minimal shared cell where all the beacons are transmitted. The difference between the two results highlights the difference between the two implementations of the protocol stack, and also the need for better modelling of the capture effect, which is future work.

The trace-based simulations for this particular evaluation is 1,200 times faster than the experiments. It takes approximately 1.5 min to execute 300 simulations on the machines in the computing cluster. For the experiments, 1,800 min, or 30 hours, are required at least: 6 min × 100 experiments × 3 parameters.

6 | CONCLUSION

In this paper, we propose trace-based simulation for 6TiSCH. Through evaluation, we confirm that the 6TiSCH Simulator can yield realistic results using connectivity traces, with far less time than experiments using hardware. We plan to see how correctly the simulator behaves with traces of a longer time, where link quality changes are observed, and with sub-GHz connectivity traces. In addition, we will evaluate the trace-based simulation in terms of application-level metrics such as throughput.

References

1. Vilajosana X, Watteyne T, Chang T, Vučinić M, Duquennoy S, Thubert P. IETF 6TiSCH: A Tutorial. *IEEE Communications Surveys & Tutorials* 2019.

2. Noble BD, Satyanarayanan M, Nguyen GT, Katz RH. Trace-based mobile network emulation. *ACM SIGCOMM Computer Communication Review* 1997.

3. Marchiori A, Guo L, Thomas J, Han Q. Realistic Performance Analysis of WSN Protocols Through Trace Based Simulation. *Proceedings of the 7th ACM workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks* 2010: 87–94.

4. Jacobsson M, Rohner C. Comparing wireless flooding protocols using trace-based simulations. *EURASIP Journal on Wireless Communications and Networking* 2013; 2013(1): 169.

5. Lamela V, Fontes H, Oliveira T, Ruela J, Ricardo M, Campos R. Repeatable and Reproducible Wireless Networking Experimentation through Trace-based Simulation. *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* 2019: 53–58.

6. Qiu L, Bahl P, Rao A, Zhou L. Troubleshooting Wireless Mesh Networks. *ACM SIGCOMM Computer Communication Review* 2006.

7. Cerpa A, Busek N, Estrin D. SCALE: A tool for simple connectivity assessment in lossy environments. *UCLA Technical Reports* 2003.

8. Werner-Allen G, Swieskowski P, Welsh M. Motelab: A wireless sensor network testbed. *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks* 2005.

9. Munoz J, Rincon F, Chang T, et al. OpenTestBed: Poor Man's IoT Testbed. *IEEE INFOCOM - CNERT : Workshop on Computer and Networking Experimental Research using Testbeds* 2019.

10. Brun-Laguna K, Minet P, Watteyne T, Gomes PH. Moving Beyond Testbeds? Lessons (We) Learned about Connectivity. *IEEE Pervasive Computing* 2018; 17(4): 15–27.

11. Municio E, Daneels G, Vicinic M, et al. Simulating 6TiSCH networks. *Transactions on Emerging Telecommunications Technologies, Wiley* 2019; 30(3).