INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2024, VOL. 70, NO. 4, PP. 1071-1079
Manuscript received October 15, 2024; revised October 2024. doi: 10.24425/ijet.2024.152362

Bit-exact simulation of 6TiSCH networks

Mateusz Kubaszek, Jan Macheta, Łukasz Krzak, and Cezary Worek

***Abstract*—This article presents an approach to building a bit-exact simulation environment, capable of handling complex wireless communication stack such as 6TiSCH with enough performance, to be useful in large scale network topologies. The architecture of the simulator and the stack is presented along with the discussion of key features and design choices. The results of the simulation are compared to the results of extensive testbed experiments, which conclude that the designed tool can be used for practical design and verification of large distributed IoT applications.**

***Keywords*—6TiSCH; network simulation; digital twin; IoT**

I. Introduction

IN recent years, we are facing a new wave of innovation leading to the development of a branch of technology called IIoT (Industrial Internet of Things) [1]. It shapes the ways of connecting and exchanging data in extensive systems and provides tools for advanced analyses. Optimization of complex production processes requires collecting and transmitting large amounts of information. IIoT provides new solutions in the form of intelligent, wireless sensor networks and is the binder of the so-called Industry 4.0. New technologies in the field of mass wireless connectivity offer new opportunities as they become the missing link between massive amounts of data sources and cloud computing centers.

Industrial applications require high reliability at a reasonable cost, hence the great interest in low power wireless systems [2][3]. In wireless networks, reliability, understood as uninterrupted availability of devices and meeting the QoS (Quality of Service) requirements, can be effectively delivered through the use of the Time Slotted Channel Hopping (TSCH) technique. This technique allows to achieve very high percentage of successful transmissions of even 99.9995% [4]. Modern applications require also high interoperability. One of the important efforts towards achieving compatibility between systems at the transport protocol level was undertaken by the IETF and resulted in the publication of the 6LoWPAN standard [5] [6]. The step that integrated both of these approaches was the adaptation of the 6LoWPAN stack to the TSCH extension as defined in the IEEE 802.15.4e standard, done by the IETF 6TiSCH working group which resulted in the publication of [7].

While being reliable, efficient and highly scalable [8], 6TiSCH networks are relatively complex, especially when deployed in potentially large topologies with hundreds of nodes. One of the key tools that allows to efficiently develop applications using 6TiSCH protocol stack in a reliable manner is an efficient and credible network simulator.

In this paper we present the approach that led to the development of such a network simulator, which can effectively emulate complex wireless distributed systems using 6TiSCH communication protocol, acting as a digital twin. The simulator uses custom 6TiSCH implementation called embeNET [9] running within the OMNeT++ discrete event simulation framework [10]. We also present the results of the comprehensive study of a 6TiSCH-based network operating in 2.4 GHz frequency band, using the Bluetooth LE 1Mbit PHY, that was used to verify and benchmark the simulator. An important aspect of the presented simulator is its bit-level accuracy, which in our case means that the simulated devices exchange exactly the same packet data (down to a bit level) and run the same network and application-level source code, built for the simulator.

II. The 6TiSCH communication stack

6TiSCH combines the 6LoWPAN with IEEE802.15.4e-TSCH MAC and Physical (PHY) layers. While challenges regarding efficient link layer resource allocation persisted, the introduction of the 6P/6top framework provided a solution [11]. This framework enabled the application of various TSCH timeslot allocation techniques. Additionally, 6TiSCH addressed node association and connection maintenance issues. The Minimal Scheduling Function (MSF) was also introduced as a straightforward decentralized timeslot allocator, enabling basic initial communication [12]. The 6TiSCH stack conducted by the IETF organization is currently in concluded state.

Nodes within 6TiSCH networks follow a scheduled pattern that is mutually established between nodes for communication. All nodes are synchronized below millisecond precision. Within the schedule the nodes allocate slots for communication. Each slot is determined by Slot Offset and Channel Offset. Communication cells are organized within recurring slot frames (Fig. 1). A cell can be bidirectional, which means that there can be more than one node sending packets within a single cell.

The article presents the results of R&D work carried out as part of the project entitled ”Development of an innovative wireless communication platform for the Internet of Things applications” (RPMP.01.02.01-12-0261/1), co-financed by the European Union under sub-measure 1.2.1 ”Research and development projects of enterprises”, Regional Operational Programme for the Małopolska Region 2014-2020.

Authors are with AGH University of Krakow, Faculty of Computer Science, Electronics and Telecommunications, Institute of Electronics, Poland (e-mails: kubaszek@agh.edu.pl, https://orcid.org/0000-0002-8670-0088; macheta@agh.edu.pl, https://orcid.org/0000-0002-0290-0606; lkrzak@agh.edu.pl, https://orcid.org/0000-0001-8139-829X; worek@agh.edu.pl, https://orcid.org/0000-0003-3741-5501).

The TSCH scheme must provide uniform channel randomization. The channel selection follows the formula below (1). Provided that the number of channels and slots count in slotframe are mutually prime numbers, the requirement for uniform randomization is satisfied.

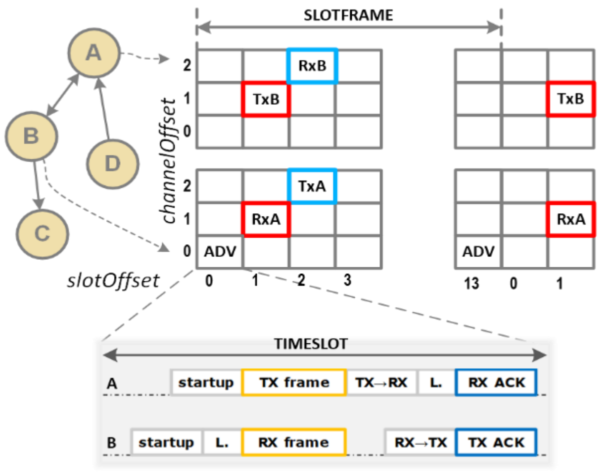

Fig. 1. TSCH timeslot organization diagram

$$\text{chan} = \text{F}(\text{channelOffset} + ASN)\text{mod}(\text{nFreq}) \qquad (1)$$

Where F is a function that maps *channelOffset* to set of channels, *ASN* (Absolute Slot Number) is a monotonic slot counter and *nFreq* is the number of available channels.

There are a couple of cell types that can be distinguished:

- Advertisement Cells. Predefined bidirectional cells within a slotframe that serve the purpose of network joining and parent selection.
- Autonomous Cells. These unidirectional cells are designated for packet transmission when managed cells are not assigned, such as during the association phase.
- Managed Cells. Unidirectional cells employed for managing data traffic. Negotiated on demand. In our implementation one obligatory unidirectional cell is allocated during joining project. The final count of managed cells adapts to the amount of traffic between nodes.

To initiate association with the network, a node receives a Beacon packet containing Information Elements (IE) crucial for synchronization with the slot frame. The synchronization is maintained by analyzing packet reception and acknowledgment times throughout the node's operation. Subsequently, the node undergoes a joining process facilitated by the preferred Join Proxy node [13]. The next step is the participation in routing procedures defined by RPL ROLL protocol [14]. Initially, the node listens for DIO (Destination-Oriented Directed Acyclic Graph Information Object) packets to select a parent node. (These DIOs and Beacons are periodically transmitted over a shared advertisement cell. Note that all nodes within the network work in non-storing mode and the DODAG Root role is assigned to Border Router device. Therefore, every node, while connected to the network, transmits its own Destination Advertisement Object (DAO) packets to the DODAG Root device. DAO packets are acknowledged using DAO-ACK packets to verify the DODAG Root presence within the network. DODAG Root stores information from all DAOs to obtain routing information to construct source routes to all devices on the network.

III. Related Works

Several network simulators, capable of simulating 6TiSCH or similar communication protocols have already been presented in the literature.

- TSCH-Sim simulator [15] is a modular, open-source project incorporating the 6TiSCH model to simulate network performance. It offers almost linear increase in simulation time with increasing number of nodes. The high scalability makes the simulator useful even with thousands of nodes. Validation was carried out by comparison of various parameters with COOJA and IETF 6TiSCH simulator. Additionally, it features an energy consumption model which was laboratory tested using the TI CC2650 SoC.
- IETF 6TiSCH simulator [16] offers similar functionalities and performance as the abovementioned TSCH-Sim. Simulated is only a 6TiSCH model not a full implementation running on real devices. Authors also do not compare results with metrics from real deployments. Validation is only performed by comparison with OpenSim tool on relatively small networks with the number of nodes less than ten.
- SEmulate [17] uses Hardware-in-the-Loop (HIL) approach. The simulated stack down to the MAC layer may cooperate with real hardware via a custom protocol. SEmulate is capable of simulating the 6TiSCH network. However, such a solution introduces an additional problem concerning the 6tisch network, where nodes are synchronized with high accuracy. Any additional intermediate layer between MAC and PHY can cause disruptive delay.
- OpenSim [18] uses the open source OpenWSN 6TiSCH stack implementation in C language and links it with a custom event simulator implemented in Python. One of the interesting features of this tool is real-time simulation with possible simultaneous interactions with physical devices and networks. The main drawback is the scalability - the practical use of this tool ends with small topologies of less than 10 nodes.
- COOJA simulation tool [19] was developed with a similar approach to OpenSim. The Contiki operating system, which integrates the 6TiSCH stack, can be run in conjunction with COOJA simulator implemented in Java. The simulator provides hardware emulation, channel model and a rich graphical environment to explore many aspects of simulation. This solution, however, like the OpenSim does not give satisfactory simulation speed.
- TOSSIM [20] is an interesting example of a complex approach to network stack simulation using the TinyOS embedded operating system. The most interesting part is the linear scalability of the simulation time with increasing number of nodes. However, the Authors claim that simulation reaches the real-time speed already at about 32 nodes which makes it cumbersome to simulate networks consisting of hundreds of nodes.

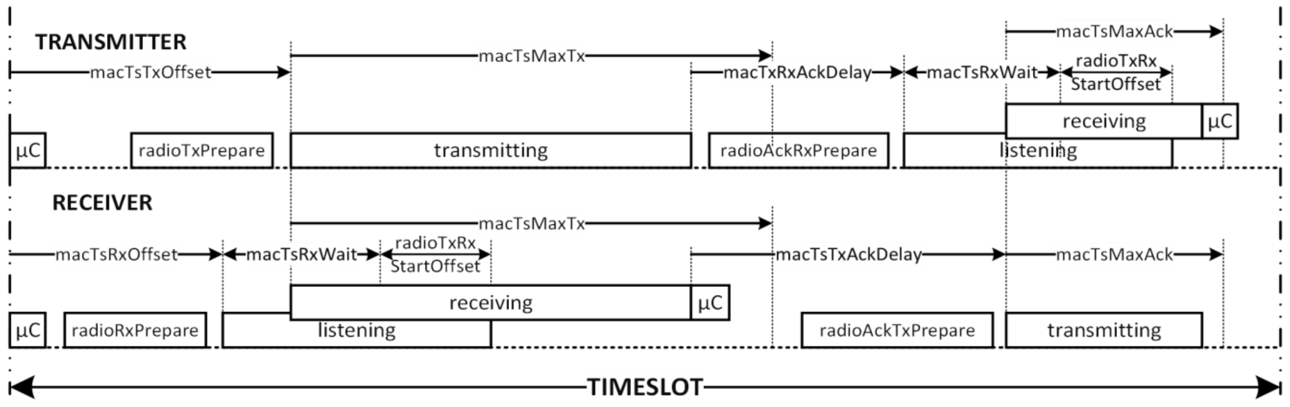

Fig. 2. TSCH slot timings with MCU and radio activity

IV. Comparison of COOJA and OpenSIM

Our solution combines the running of the actual stack implementation code along with hardware emulation of the radio transceiver and rich radio propagation modeling. In that regard our approach is similar to COOJA and OpenSim. However, there are some important differences that we want to point out:

- The 6TiSCH stack is fully implemented within the simulator, which provides bit-exact simulation of the communication protocol Exactly the same code runs in physical devices. The stack code architecture is completely event-driven which allowed us to run it within a popular OMNET++ discrete event simulator framework. This framework provides important features such as modularity, network description and configuration language called NED, and an effective mechanism for scheduling, managing and executing events.
- Despite its bit-exact nature, the achievable simulation time for 1,000 nodes is about 2-7 times faster than real time on a modern computer which makes it still usable at such scale.
- We employ standard log distance path loss model for radio channel modelling. The characteristics of all links are individually calculated with path loss, fading and mutual interferences.
- The hardware emulation includes complex model of the radio transceiver, including details such as crystal accuracy and time drift, switching timings and more.
- Our tool features fast data acquisition using SQLite database, which allows to gather important metadata during the simulation from all the nodes in a very flexible manner.

V. 6TiSCH stack configuration

6TiSCH can run on top of several physical layers (PHYs). In our study Bluetooth Low Energy (BLE) 1Mbit PHY was used instead of the widely used IEEE 802.15.4 250 kbps DSSS-OQPSK-250 PHY layer for the 6TiSCH stack. We justify the choice with the high popularity and low price of BLE-compatible radio chipsets. Moreover, most of them are compatible with 2Mbit or long range PHY which gives more options for compromise between the range and speed of radio communication. Main differences between aforementioned PHYs are presented in the Table I.

Table I
Short comparison of BLE 1Mbit PHY and IEEE 802.15.4 250 kbps DSSS-OQPSK-250 PHY

Parameter	BLE 1Mbit	IEEE 802.15.4 DSSS-OQPSK-250
modulation scheme	2GFSK	DSSS-OQPSK
bitrate	1000 kbps	250 kbps
channel width	2 MHz	5 MHz
channel count in unlicensed 2.4GHz band	40	16
typical slot length	5000 us	10000 us
typical transceiver sensitivity	~-95dBm	~-100dBm

The use of BLE 1Mbit PHY dictates a specific set of time parameters that composes the MAC timings set (Fig. 2 and Table II).

Table II
Short comparison of BLE 1Mbit PHY and IEEE 802.15.4 250 kbps DSSS-OQPSK-250 PHY

Time parameters (MAC PIB)	Value [μs]
macTsLenght	5000
macTsTxOffset	1700
macTsRxOffset	1200
macTsMaxTx	2000
macTsMaxAck	1000
macTsRxWait	1000
macTsAckWait	300
macTsTxAckDelay	1000
macTsRxAckDelay	850

Table III presents important configuration parameters that have a significant impact on the 6TiSCH communication performance.

Table III
Important stack configuration parameters

Parameter	Value
active time slots count in slotframe[a]	21
slotframe length [slots]	61
mean Beacon sending interval	every sixth slotframe
mean DIO sending interval	every sixth slotframe
advertising channels (BLE assignment)	37, 38, 39
data channels (BLE assignment)	0-36

[a] during first 21 slots in slotframe node is active, during the remaining time node is inactive

VI. Hardware

The hardware used in the testbed were custom and stock evaluation boards incorporating the popular nRF52832, nRF52833 and nRF52840 SoCs from Nordic Semiconductor. Radio output power in all devices was set to 4dBm . Any differences in antenna and radio front-end performance can be considered negligible in this experiment. The selected hardware imposes an additional set of parameters that limits timings in the MAC layer (Table IV).

Table IV
Important stack configuration parameters

Hardware related parameters	Value
sensitivity (1% of PER on 32B of PSDU)	-93dBm*
max output power	4dBm
system clock accuracy	+-20ppm
radio clock accuracy	+-10ppm
radioTxRxStartOffset	40µs*
radioTxPrepare/radioRxPrepare	400µs*
radioAckTxPrepare/radioAckTxPrepare	40µs*
sleep current	5µA
tx current	17.5mA
listening current	13mA
required signal dynamics	15dB

[a] measured

VII. The internals of the 6TiSCH simulator

Figure 3 presents the overall architecture of the 6TiSCH simulator. The stack provides an API for connection management and data transmission that can be used by the final Application. The stack also relies on a platform specific port. In real-world deployments this port consists of drivers for hardware peripherals such as timers and radio transceiver. However, when running within the simulation, this port emulates the required functions with the use of the OMNeT++ features such as messaging. Within this part of the simulator, we also implement the radio propagation model and emulate transceiver's frame handling engine.

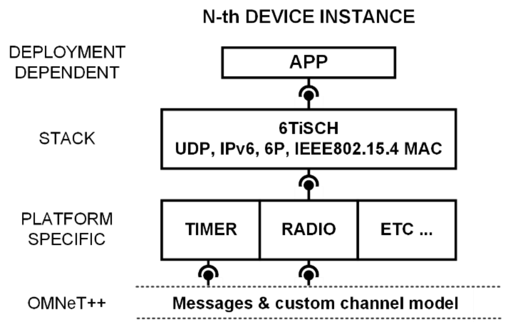

Fig. 3. Block diagram illustrating the 6TiSCH simulator architecture

The stack itself is implemented according to the principles of event-driven architecture. All of the procedures within the stack are invoked by underlying timer and radio transceiver. This gives trouble-free integration with a discrete event simulator, such as OMNeT++. However, discrete event simulation neglects one aspect – the event code execution time. When working with time critical code, which is the case for highly synchronous 6TiSCH protocol, this may lead to inconsistent simulation results. The solution turned out to be measuring the execution time of time-critical procedures for each covered hardware platform and providing that time within the port. Once the stack has these parameters provided, it is able to schedule time-critical procedures a certain amount of time in advance, thus compensating for the additional required processing time on a given platform.

The employed physical channel is modelled as a log distance path loss model which is an extension to the Friis free space model [21]. This model is relatively simple and adaptable to many propagation environments from urban spaces to building interiors. The log distance path loss formula (2):

$$\mathrm{L}(d) = \mathrm{E} * 10 * \log_{10}\frac{d}{\mathrm{D}} + \mathrm{L}_{\mathrm{Friis}}(\mathrm{D}) + \chi_{\mathrm{t}}[\mathrm{dB}] \qquad (2)$$

Where E[dB] is the environment dependent path loss exponent, D is the close-in reference distance, LFriis[dB] is the Friis path loss for an arbitrary distance D and χt[dB] is a zero-mean Gaussian distributed random variable with σ expressed in dB. Example results of the applied propagation model are presented in the Fig. 4.

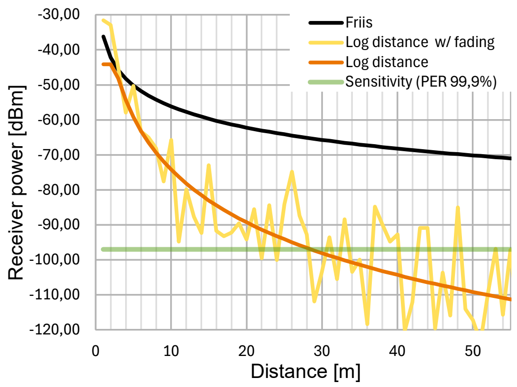

Fig. 4. Radio signal attenuation for the propagation models discussed

The radio channel model interacts with the transceiver emulator. Fig. 5 shows the transceiver model used. It corresponds closely to the state diagrams contained in the reference manuals of many radio transceivers, since all of them operate on a similar principle. The same general operating principle is emulated by our emulator.

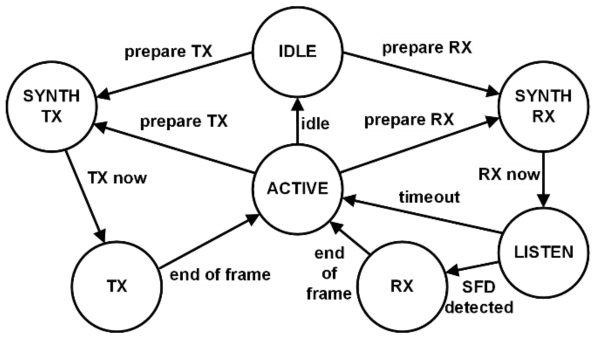

Fig. 5. The state machine emulating a radio transceiver

Key features of radio transceiver emulator:

- The simulation framework is aware of frame transmission duration based on its length and bitrate.
- The transceiver model uses a custom PER (Packet Error Rate) vs Input Power curve, measured in laboratory conditions for

the covered hardware platforms, which models the transceiver sensitivity. A Similar approach is described as Pister-Hack curve [16][18][19].

- Emulation of frame detection mechanisms. Probability of reception of frame synchronization header with SFD (Start of Frame Delimiter) and PSDU (PHY service data unit) are calculated separately.
- Monitoring of S/I (signal-to-interference ratio) during reception to make better decisions about frame collisions. If radio frames transmitted in the same channel overlap a collision occurs. However, if sufficient S/I ratio is maintained, the stronger packet can still be received. Including this effect adds another level of detail to the simulation, increasing credibility of the results.

One important aspect of the implemented radio channel model which heavily affects the performance is the preselection of viable communication links. In general, when simulating large, distributed topologies, every transmitted radio packet will reach many receivers. Calculating the probability of correct packet reception at many receivers is time consuming. It is thus important to limit the receivers, eliminating nodes that due to range of communication are almost certainly out of communication range. . For static topologies this qualification is done once at simulation startup and provides great performance optimization. The fading effect is applied on every transmission separately.

The hardware, that was tested was characterized and the resulting parameters (see Table IV) were applied to the transceiver emulator.

One additional feature of the developed simulation environment is the possibility of using a high-entropy pseudorandom generator as the randomness source for all entities. This causes the simulator to generate exactly the same results in specific runs, which is an invaluable feature in the debugging process.

As shown in figure 3 the user's application is also included in simulation. In many cases the application utilizes the stack API only which makes it easily portable across the simulation framework and the target physical device.

VIII. Deployment in a Test Bed

In order to verify the simulation engine, we established a testbed consisting of 15 battery powered nodes, 1 root node and RPI3-based border router, as shown in Fig. 6.

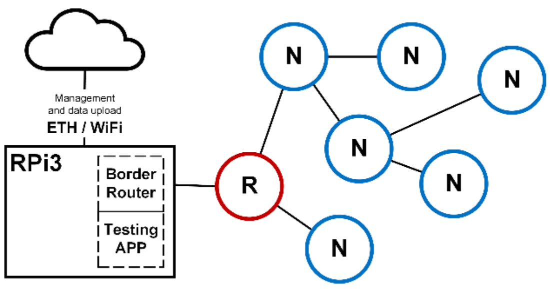

Fig. 6. Illustration of the testbed setup

The nodes were placed within an industrial environment. The test facility construction and finishing materials were mainly metal elements, including doors. Three scenarios were conducted:

A. *Whole facility*. Nodes were spaced throughout the facility, including 3 nodes placed outside. This gave relatively large link distances. The spacing was one node per adjacent room or stairwell separated by metal doors.

B. *Crowded social space*. Nodes were set up within an elongated social room with several small rooms spread along it. Border router was at the far end. The length of the room social area was about 60 meters.

C. *Production area*. 5 rooms separated by doors, extending about 50 meters total in length.

IX. Testing Entities

For every testbed deployment, one 6-hour run was conducted during which several testing entities were active. A test entity is an application that runs in the system and benchmarks it. Note that all data was collected by the network itself, so no additional out-of-band communication mechanisms were used. As all devices in the network are synchronized with hundreds of microsecond accuracy, all processes in all connected devices may be precisely timestamped. Hence every packet has its time-of-flight precisely calculated with a resolution of ASN (absolute slot number), that was set to 5ms. Every testing entity implements packets counters, so the number of lost radio packets can be determined.

Figure 7 presents the testing entities' activity during the whole test run.

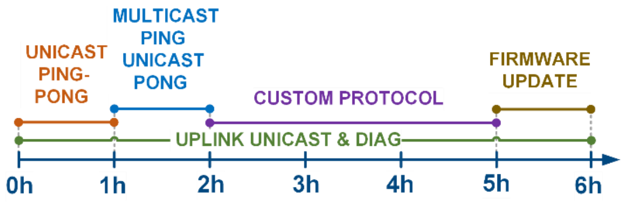

Fig.7. A chart showing the activity of testing entities in a test run

Tests were conducted on the following network operating parameters:

- *Topology stability and coverage*. Examination of how fast network becomes fully operable, parent selection, mutual range and link metrics for all nodes.
- Uplink unicast traffic. Each node generates packet to one common data sink. Each data packet is generated with uniformly distributed delay in range <15s, 45s> and carries 28 bytes of data. Moreover, diagnostics service is running generating constant uplink unicast traffic.
- *Unicast ping-pong (1h)*. A packet source being outside of the network generates individual packets to all nodes in the network. Target node is selected randomly from the connected ones. 24B ping data packet is generated with uniformly distributed delay in range <1s, 3s>. Nodes immediately respond to ping packet with 28 bytes long pong packet.
- *Multicast ping – unicast pong (1h).* A packet source being outside the network generates multicast packets to all nodes within the network with a uniformly distributed delay value in range <15s, 45s>. Every node responds with a pong packet with a uniformly generated delay in range <0s, 5s> which is

a way to offload the relatively high peak reply traffic generated in a short period of time in the network.

- *Firmware update (1h).* During the test the task is to perform the transmission of 10 kB of data to all nodes in the network, simulating firmware update process. Data bulk is sent using the multicast traffic . Data completion and consistency is checked.
- *Custom protocol (3h).* During the inactive part of the superframe, custom protocol was active. Its operation is out of scope of this article.

Exactly the same testing scheme as conducted in real-life deployment was simulated in the simulation environment. However, the simulation did not reproduce the exact distribution of the physical devices in space. For all simulation runs the topology was created as a 5x3 grid with randomized edge distance with uniform distribution in range <3.75m, 11.25m> for scenario A, <5m, 15m> for scenario B and <7.5m, 22.5m> for scenario C. The root node in all scenarios was placed in the corner of the network.

X. Outcome of the Study

After analyzing the results from scenario A, it became clear that the network nodes were too far apart. The network was operating in edge conditions in which the connections between nodes were too weak or even non-existent for several nodes. As a result, run A was excluded from further analysis.

All simulation metrics for scenarios B and C were calculated as averages of metrics from 10 random runs.

Figure 8 shows the dependence of PDR (Packet Delivery Ratio) on RSSI (Received Signal Strength Indication) obtained from the testbed and from two randomly selected simulation runs. The first interesting observation was that the radio system performed much worse in the field than during the laboratory tests. At the testing facility, the effective sensitivity curve turned out to be shifted and more flattened. The estimated sensitivity was -75dBm, not -95dBm, as confirmed by prior laboratory measurements. To solve this problem in the simulation, without changing the parameters of the radio model, we set the standard deviation of the fading to 10dB. In this way, the simulated performance of the radio was close to that measured during field tests.

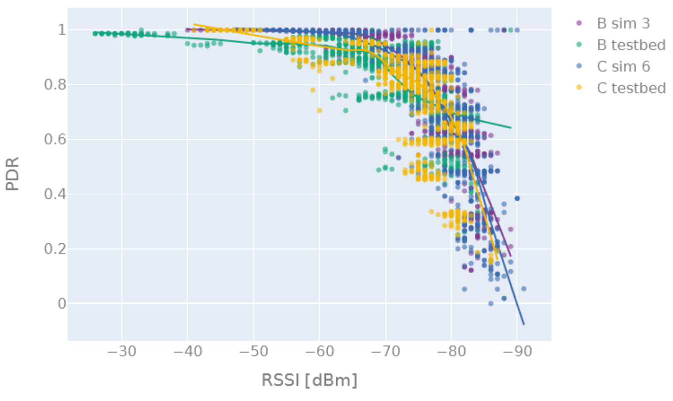

Fig. 8. Simulated and real-life PDR vs RSSI diagram

During both runs B and C, the network became fully operational, all devices connected and had a stable connection over the network. The same results were obtained during the simulation. However, as can be seen from Table V, there were some significant differences between the simulation and actual results in terms of network formation speed.

Table V
Topology Formation Metrics

Run	Joined	First [s]	Median [s]	Last [s]
B testbed	15	82	526	848
B simulation	15	49	88	298
C testbed	15	408	540	919
C simulation	15	48	128	593

Figure 9 shows the graphs of nodes joining the network for scenario C. The testbed experiments gave similar results, but with a significant time offset. This is due to the fact that the network took some time to disconnect from the previously running network run before launching. This was our oversight, and unfortunately, for organizational reasons, we could not repeat the experiment. However, the dynamics of nodes connecting the network were similar in all cases, most nodes attached quickly except for the last few.

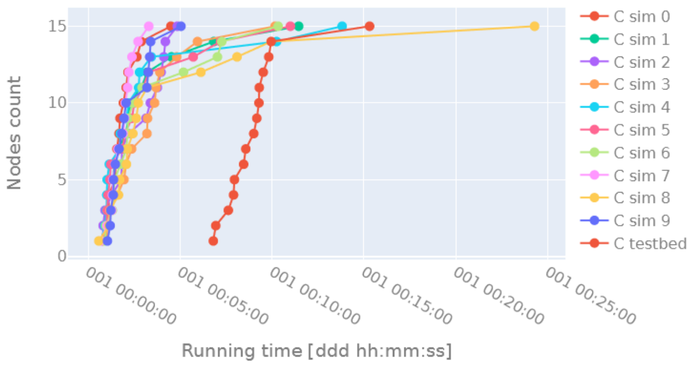

Fig. 9. Number of nodes joining the network vs running time

Figure 10 shows the network topologies obtained from real runs and from two randomly selected simulations. The main difference between scenario B and C is the depth of the network, which affects the performance of data traffic. We decided not to create topology metrics because two specific real-world runs were not sufficient to collect a representative amount of data. It turned out that in testbed launch C the network adopted a specific topology and this had to do with the location of the nodes. Traffic from almost the entire network is routed through a single node 2 (Fig. 10b). This was evident in the results, where the average packet delivery time was slightly longer than expected.

TABLE VI
UPLINK UNICAST TRAFFIC METRICS

Run	Sent	TOF min [s]	TOF median [s]	TOF max [s]	Lost	PDR [%]
B testbed	10350	0.005	0.21	52.18	44	99.6%
B simulation	10750	0.005	0.20	13.52	1	100.0%
rel. diff.	4%	0%	-6%	-74%	-99%	0.4%
C testbed	10343	0.005	0.45	41.31	105	99.0%
C simulation	10515	0.005	0.27	52.57	103	99.0%
rel. diff.	2%	0%	-40%	27%	-2%	0%

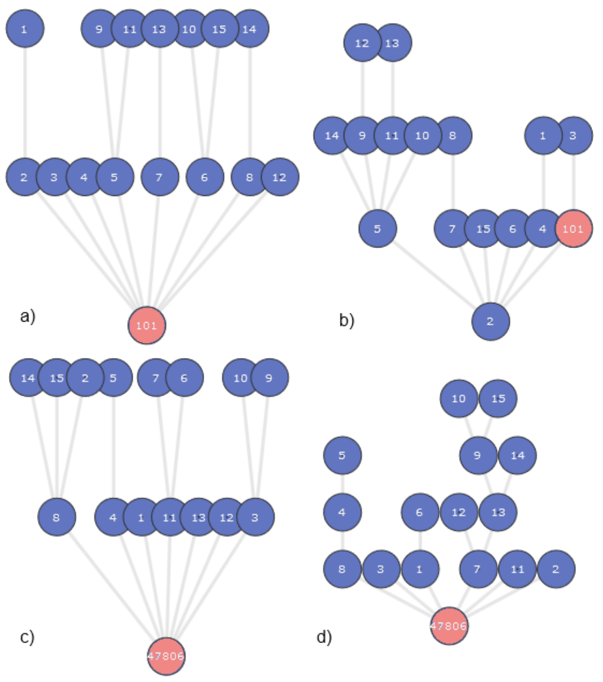

Fig. 10. Topology snapshot from: a) scenario B in testbed, b) scenario C in testbed, c) scenario B in simulation , d) scenario C in simulation

Table VI summarizes the metrics for unicast uplink traffic. In terms of PDR, the data obtained from the simulation and the testbed are similar. In terms of median packet transit time, the simulation performed better - in the simulation, on average, packets were delivered faster. This is due to the specific topology formed by the real network. Figures 11 and figure 12 show the difference in round-trip packet flight time (RTT) between real scenarios B and C. In such a small network, the impact of the shape of the topology is visible as clustered ranges of delay time.

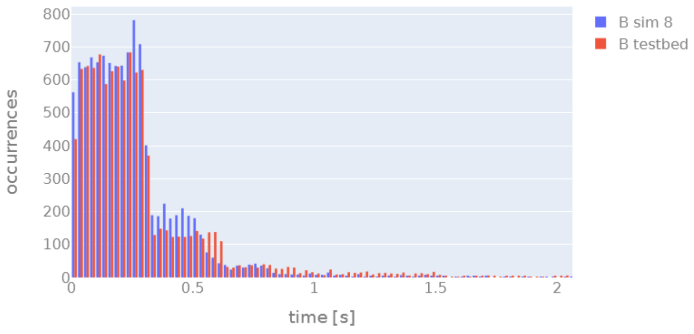

Fig. 11. Histogram of time-of-flight of uplink unicast data packets (testbed and randomly selected simulation of scenario B)

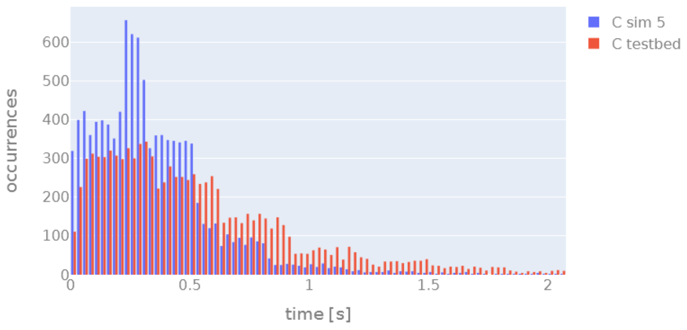

Fig. 12. Histogram of time-of-flight of uplink unicast data packets (testbed and randomly selected simulation of scenario C)

The unicast ping-pong performance data is shown in Table VII. As with the unicast uplink traffic study, the differences in PDR are marginal. Since the RTT is split into downlink and uplink components, it is clear that the main difference between the simulation and the actual deployment is the uplink round-trip time. The uplink transit time in scenario B in the real-world test was better than in the simulations compared to scenario C, where the simulation performed better. As with uplink unicast traffic, the ping-pong delay histogram reveals topology-dependent differences in RTT (Fig. 13 and Fig. 14).

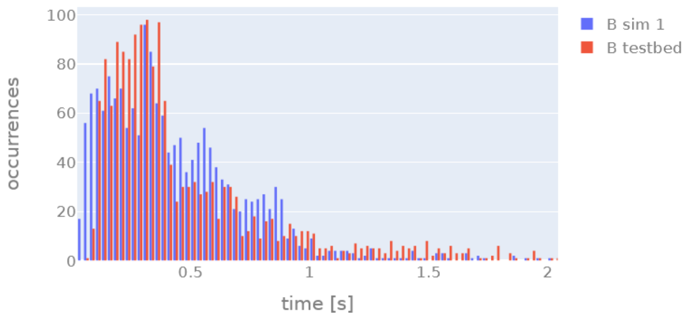

Fig. 13. Histogram of RTT of unicast ping-pong queries (testbed and randomly selected simulation of scenario B)

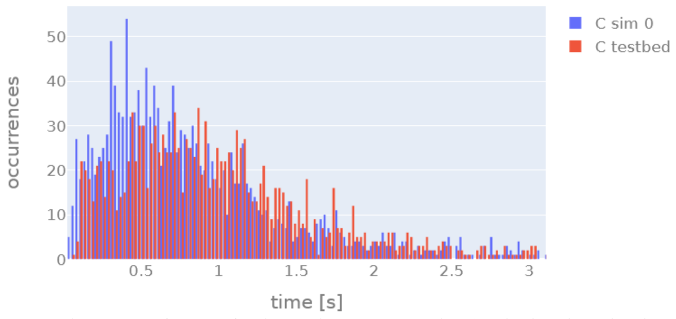

Fig. 14. Histogram of RTT of unicast ping-pong queries (testbed and randomly selected simulation of scenario C)

TABLE VII
UNICAST PING-PONG TRAFFIC METRICS

Run	Sent	Time-of-flight [s]									Round-trip	
		Downlink			Uplink			Round-trip				
		Min	Median	Max	Min	Median	Max	Min	Median	Max	Lost	PDR
B test	1723	0.020	0.22	2.13	0.02	0.09	42.15	0.07	0.35	43.24	15.00	99.1%
B sim	1725	0.010	0.20	2.38	0.01	0.21	11.11	0.03	0.39	11.85	1.30	99.9%
rel. diff.	0%	-50%	-11%	12%	-33%	146%	-74%	-57%	11%	-73%	-91%	1%
C test	1543	0.020	0.36	3.71	0.03	0.37	49.80	0.07	0.92	52.73	29.00	98.1%
C sim	1736	0.010	0.30	4.77	0.01	0.18	45.22	0.04	0.57	47.24	94.50	94.6%
rel. diff.	12%	-50%	-15%	29%	-57%	-52%	-9%	-48%	-38%	-10%	226%	-4%

TABLE VIII
MULTICAST DOWNLINK – UNICAST UPLINK TRAFFIC METRICS

Run	Unique deliveries	Time-of-flight [s]									Round-trip	
		Downlink			Uplink			Round-trip				
		Min	Median	Max	Min	Median	Max	Min	Median	Max	Lost	PDR
B test	1770	0.020	0.21	0.81	0.005	0.21	35.71	0.035	0.42	36.03	57	96.8%
B sim	1761	0.012	0.20	0.91	0.005	0.20	4.04	0.027	0.39	4.33	12	99.3%
rel. diff.	-1%	-43%	-5%	13%	0%	-5%	-89%	-24%	-6%	-88%	-79%	3%
C test	1665	0.030	0.27	1.42	0.005	0.87	9.13	0.090	1.20	9.65	169	89.8%
C sim	1743	0.012	0.27	3.07	0.005	0.27	35.62	0.030	0.58	36.18	109	93.7%
rel. diff.	5%	-60%	-1%	117%	0%	-68%	290%	-67%	-52%	275%	-35%	4%

Table VIII show the metadata of multicast traffic. The same observation was made - for Scenario C, unicast uplink traffic performed much better in the simulation. Nevertheless, the most important parameters for the application, PER and multicast downlink transit time, were very similar for both scenarios. Delivery metrics were also generated. Table IX shows the number of devices the packet reached. The worst, average and best of all transmissions are given. In Scenario C in the real-world test, it happened that one of more than a hundred multicast packets did not reach the network correctly, hence only one device was reached instead of all of them.

TABLE IX
MULTICAST PACKET DELIVERY COUNT PER TRANSMISSION

Run	Worst delivery count	Avg delivery count	Best delivery count
B testbed	15	82	526
B simulation	15	49	88
C testbed	15	408	540
C simulation	15	48	128

CONCLUSION

When developing scalable, distributed applications featuring complex communication protocols it is crucial to be able to reliably simulate them, prior to any large-scale deployments. A simulation tool fit for that purpose should allow to run exactly the same code in the simulation as in the physical device and include as many important model details as possible.

In this article we've presented a tool that follows these principles, by merging a portable, event-driven 6TiSCH stack implementation with a popular discrete event simulator (OMNeT++), detailed radio transceiver emulation and proper radio channel modeling. By compiling the actual stack code directly with the simulation engine, avoiding graphical user overhead and introducing several important optimizations the tool is able to efficiently simulate networks consisting of even thousands of nodes. It became an important asset in our professional work, in the process of designing, prototyping and deployment of 6TiSCH-based radio communication systems.

The article also presents an effort made to validate the simulator using an extensive set of test entities that were run in a physical test bed consisting of 15 nodes. The same entities were run in the simulation and results were compared. While there are some differences, in general we've received very similar network performance in simulation and the real world, as confirmed by the values of the metrics presented.

We can thus conclude that the applied methodology has great potential to be verified in much larger testbeds, which we plan to do in the near future.

REFERENCES

[1] S. Al-Sarawi, M. Anbar, R. Abdullah, A Al Hawari, "Internet of things market analysis forecasts," Proceedings of the World Conference on Smart Trends in Systems, Security and Sustainability, WS4, pp. 449—453, 2020. A. Author, "Book style with paper title and editor," in Title, 1nd ed. vol. 1, C. Editor, Ed. City: Publisher, pp. 10–50, 2017. https://doi.org/10.1109/WorldS450073.2020.9210375

[2] D. Dujovne, T. Watteyne, X. Vilajosana and P. Thubert, "6TiSCH: deterministic IP-enabled industrial internet (of things)," in IEEE Communications Magazine, vol. 52, no. 12, pp. 36-41, December 2014. https://doi.org/10.1109/MCOM.2014.6979984

[3] X. Vilajosana, T. Watteyne, M. Vucinic, T. Chang, K. S. J. Pister, "6TiSCH: Industrial Performance for IPv6 Internet-of-Things Networks," in Proceedings of the IEEE, vol. 107, n. 6, pp. 1153—1165, 2019. https://doi.org/10.1109/JPROC.2019.2906404

[4] K. Pister, L. Doherty, "TSMP: Time Synchronized Mesh Protocol," IASTED International Symposium on Distributed Sensor Networks, DSN, pp. 391—398, 2008.

[5] Y. Tanaka, K. Brun-Laguna, T. Watteyne, "Demo: Simulating a 6TiSCH Network using Connectivity Traces from Testbeds," in Transactions on Emerging Telecommunications Technologies, vol. 30, n. 3, 2019. https://doi.org/10.1109/INFCOMW.2019.8845200

[6] T. Watteyne, X. Vilajosana, B. Kerkez, ~et al., "OpenWSN: A standards-based low-power wireless development environment," European Transactions on Telecommunications, vol. 23, n. 5, pp. 480—493, 2012. https://doi.org/10.1002/ett.2558

[7] X. Vilajosana, K. Pister, T. Watteyne, "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration," RFC 8180, May 2017. https://doi.org/10.17487/RFC8180

[8] R. C. A. Alves and C. B. Margi, "IEEE 802.15.4e TSCH Mode Performance Analysis," 2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Brasilia, Brazil, pp. 361-362, 2016. https://doi.org/10.1109/MASS.2016.054

[9] EmbeTech, embeNET Suite, https://embe.tech/embenet, access 15 April 2024.

[10] Cogitative Software, OMNeT++, https://omnetpp.org, access 15 April 2024.

[11] Q. Wang, X. Vilajosana, T. Watteyne, "6TiSCH Operation Sublayer (6top) Protocol (6P)," RFC 8480, November 2018. https://doi.org/10.17487/RFC8480

[12] T. Chang, M. Vucinic, X. Vilajosana, S. Duquennoy, D. R. Dujovne, "6TiSCH Minimal Scheduling Function (MSF)," RFC 9033, May 2021. https://datatracker.ietf.org/doc/rfc9033

[13] M. Vucinic, J. Simon, K. Pister, "Minimal Security Framework for 6TiSCH," online, available: https://datatracker.ietf.org/doc/draft-ietf-6tisch-minimal-security/01/

[14] R. Alexander, ~et.al., "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550, March 2012. https://doi.org/10.17487/RFC6550

[15] A. Elsts, "TSCH-Sim: Scaling Up Simulations of TSCH and 6TiSCH Networks" Sensors 20, no. 19: 5663, 2020. https://doi.org/10.3390/s20195663

[16] E. Municio, G. Daneels, M. Vucinic, ~et al., "Simulating 6TiSCH networks," Transactions on Emerging Telecommunications Technologies, vol. 30, n. 3, 2019. https://doi.org/10.1002/ett.3494

[17] S. Boehm and H. Koenig, "SEmulate: Seamless Network Protocol Simulation and Radio Channel Emulation for Wireless Sensor Networks," 2019 15th Annual Conference on Wireless On-demand Network Systems and Services (WONS), Wengen, Switzerland, pp. 111-118, 2019. https://doi.org/10.23919/WONS.2019.8795495

[18] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, K. Pister, "OpenWSN: a standards-based low-power wireless development environment," Trans. Emerging Tel. Tech., 23: 480-493, 2012. https://doi.org/10.1002/ett.2558

[19] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne and T. Voigt, "Cross-Level Sensor Network Simulation with COOJA," Proceedings. 31st IEEE Conference on Local Computer Networks, Tampa, FL, USA, pp. 641-648, 2006. https://doi.org/10.1109/LCN.2006.322172

[20] P. Levis, N. Lee, M. Welsh, D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys '03). Association for Computing Machinery, New York, NY, USA, 126–137. https://doi.org/10.1145/958491.958506

[21] M. Viswanathan, "Wireless Communication Systems in Matlab," independent publication, 2020. ISBN: 979-8648350779.