A Tutorial on Performance Evaluation and Validation Methodology for Low-Power and Lossy Networks

Kosmas Kritsis, Georgios Papadopoulos, Antoine Gallais, Periklis Chatzimisios, Fabrice Theoleyre

A Tutorial on Performance Evaluation and Validation Methodology for Low-Power and Lossy Networks

Kosmas Kritsis, Georgios Z. Papadopoulos, *Member, IEEE,* Antoine Gallais,
Periklis Chatzimisios, *Senior Member, IEEE,* and Fabrice Théoleyre, *Senior Member, IEEE,*

***Abstract*—Envisioned communication densities in Internet of Things (IoT) applications are increasing continuously. Because these wireless devices are often battery powered, we need specific energy efficient (low-power) solutions. Moreover, these smart objects use low-cost hardware with possibly weak links, leading to a lossy network. Once deployed, these Low-power Lossy Networks (LLNs) are intended to collect the expected measurements, handle transient faults and topology changes, etc. Consequently, validation and verification during the protocol development are a matter of prime importance. A large range of theoretical or practical tools are available for performance evaluation. A theoretical analysis may demonstrate that the performance guarantees are respected, while simulations or experiments aim on estimating the behaviour of a set of protocols within real-world scenarios. In this article, we review the various parameters that should be taken into account during such a performance evaluation. Our primary purpose is to provide a tutorial that specifies guidelines for conducting performance evaluation campaigns of network protocols in LLNs. We detail the general approach adopted in order to evaluate the performance of layer 2 and 3 protocols in LLNs. Furthermore, we also specify the methodology that should be adopted during the performance evaluation, while reviewing the numerous models and tools that are available to the research community.**

***Index Terms*—Low-power Lossy Networks; Internet of Things; Protocols; Algorithms; Performance Evaluation; Validation; Experiments; Simulation; Testbeds; Models**

I. Introduction

After many decades of research, wireless networks have evolved from Ad Hoc Wi-Fi technology to low-power and large-scale Wireless Sensor Networks (WSNs). This miniaturization extended de facto their usage, and enabled the design of radically new applications that follow the modern concept of Internet of Things (IoT). For instance, Smart Cities are expected to rely heavily on a myriad of devices, able to measure their surrounding environment and take decisions in order to manage the city efficiently [1], [2]. Moreover, sensors may be used for counting the number of vehicles, such to control optimally the street traffic lights and to reduce the waiting time [3].

This type of sensor networks, due to their embedded nature, often requires to operate with limited power, constrained memory and processing resources. Therefore, they require specific approaches to make the wireless network energy efficient. To do so, the wireless devices have to implement protocols to reduce their energy consumption, e.g., by turning off their radio interface for most of the time.

Furthermore, multiple environmental factors interfere with the system, while adding extra noise and promoting high variances in the communication links. Therefore, the Packet Error Rate (PER) may be high for some of the radio links, and requires to design *robust* algorithms and protocols. In particular, we have to make the system reliable even if it depends on unreliable radio links. The wireless infrastructure has to deliver most of the packets to their destinations, without duplicating them (energy efficiency), and by handling transparently the packets losses at the link layer.

Actuators may also be part of the wireless infrastructure (e.g., the heating system in smart buildings [4]). To avoid any ambiguity, we adopt here the Internet Engineering Task Force (IETF) terminology, which designates a Wireless Sensor Network as a *Low-power Lossy Network* (LLN). A LLN comprises routers, sensors and actuators, which use wireless transmissions to exchange packets, possibly via multiple hops.

The great importance of LLNs becomes clear, considering how they have affected the emergence of modern IoT applications, where information must be shared intact across different platforms, while enhancing the data from a distributed network of sensors and actuators.

Globally, the IoT paradigm encompasses a large variety of devices connected to the Internet, including amongst others: (i) multimedia objects (e.g., a video camera) that require a high throughput and, thus, rendering them unable to save energy, (ii) Radio-Frequency Identification System (RFID) and tags that enable the tracking of any object, typically within a supply chain context, (iii) small sensors and actuators, which are mostly battery-powered wireless hardware platforms.

The number of embedded devices involved in a typical LLN scenario varies from tens to thousands of nodes, which introduces further density as well as scalability issues.

Validation and verification during the protocol development has become a matter of prime importance. Envisioned

K.Kritsis is with the Research and Innovation Center Athen (ATHENA RC), Institute for Language and Speech Processing (ILSP), 15125 Marousi, Athens, Greece (e-mail: kosmas.kritsis@ilsp.gr).

G. Z. Papadopoulos is with the IRISA, IMT Atlantique, 35510, Cesson-Sévigné, France (e-mail: georgios.papadopoulos@imt-atlantique.fr).

A. Gallais and F. Théoleyre are with the ICube Laboratory, CNRS / University of Strasbourg, 67412 Illkirch, France, (e-mail: {gallais,theoleyre}@unistra.fr).

P. Chatzimisios is with the CSSN Research Lab, Department of Informatics, Alexander TEI of Thessaloniki (ATEITHE), 57400 Thessaloniki, Greece and with the Department of Computing & Informatics, Bournemouth University, BH12 5BB, United Kingdom (e-mail: peris@it.teithe.gr).

solutions must be intensively tested before being deployed in a real-world environment, by employing either simulators, emulators, or even conducting experimentations over real physical testbeds [5]. More specifically, simulators have allowed users to exploit available models (e.g., link quality, radio propagation, medium interference, topologies) in order to anticipate the behaviour of their proposals under real-life conditions. Some open testbeds have also emerged, providing access to pre-deployed low-power devices and, thus, a certain level of repeatable experimental setups [6]. However, a great number of issues, such as weather conditions, interference from other wireless technologies or even obstacles, may have an impact on the radio links among the sensor nodes. Thus, before proceeding to the experimental evaluation of a network protocol or an algorithm, apprehending the wireless links and the overall topology is an essential step [7]. It is also mandatory to consider the correct energy models since preliminary results must help end users to get a flavour of the ensued energy consumption. We here detail numerous tools that shall ease such investigations.

In order to initially validate their concepts or models, researchers must be advised of the numerous constraints that arise in order to define the scientific methodology that will lead to pertinent experimental results. Theoretical and numerical analysis, along with packet level simulation and emulation appear as essential pieces of the long-term validation of a whole solution.

In this tutorial, we review in depth various parameters that should be taken into account during such performance evaluation. We especially focus on the main networking layers of the communication stack (e.g., Medium Access Control (MAC), routing). Techniques related to the physical layer are out of the scope of this document, since the performance evaluation often uses specific hardware for the experiments, or it is conducted based on specific models. The link and network layers are rather based on protocols and algorithms, and form a specific, distinct research subject, with different tools.

Our primary purpose is to provide a tutorial both with basic and enhanced guidelines for evaluating the performance of network protocols in LLNs.

The contributions of this paper are fourfold:

1) We detail the general approach for evaluating the performance of networking protocols for LLNs;
2) We detail the methodology to be adopted, what should be measured (i.e., metrics) and what should be taken into account when evaluating the performance;
3) We provide a comprehensive view of the different models (radio communication, interference, traffic) and their typical characteristics and limitations. In particular, we provide key use cases in which the protocols should be evaluated;
4) Finally, we detail the existing tools, from theory to experimentations, and their typical utilization.

The paper is organized as follows. In Section II, we expose the suggested methodology that allows a complete and relevant performance evaluation. Section III details the main requirements of a consistent performance evaluation process, ranging the monitored parameters that are related to the accuracy of the setup (e.g., reproducibility). We then describe the various characteristics that may be present in various radio topologies (Section IV), as well as we detail the traffic characterization (Section V) and the different energy models (Section VI). We then present most of the existing tools that may be used during a performance evaluation campaign (Section VII). Finally, Section VIII discusses the current related challenges, before giving some concluding remarks in Section IX.

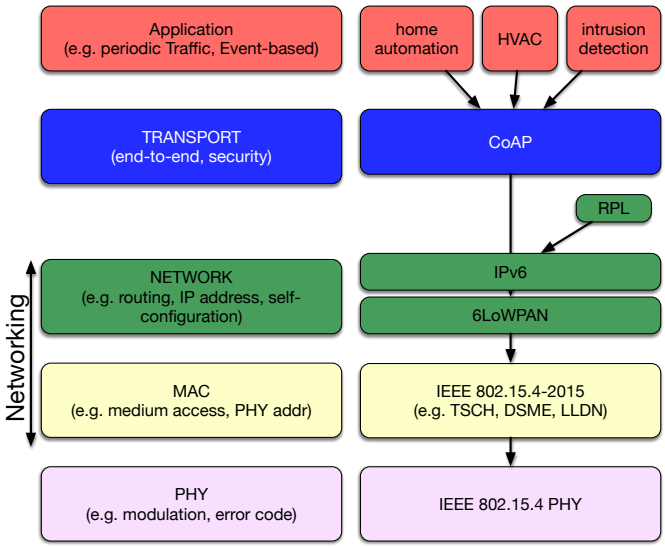


Fig. 1: Typical stack of protocols in the Internet of very low-power Things.

II. Performance Evaluation Life-circle

A LLN is a combination of software, network and embedded engineering. These fields are well defined and therefore, developers should be aware of the currently employed technologies and methods in the previously mentioned domains in order to efficiently design new solutions.

However, some of these practices should be modified in order to fit the specifications of wireless LLNs. There are various methodologies that can be adapted, concerning the development life-cycle of applications that incorporate LLN technologies.

Most LLN solutions are built upon a modular *stack* of protocols (Figure 1): the most accurate protocols / standards may be chosen before the deployment, based on the requirements of the targeted specific application. To reduce the cost of deployment, several applications (e.g., intrusion detection, heating, ventilating and air-conditioning), may operate on top of the same wireless infrastructure [8]. Transport layer solutions dedicated to LLNs include CoAP and MQTT-SN, which both rely on publish/subscribe mechanisms, while using UDP as the underlying protocol [9], [10].

Agile methodologies are particularly appropriate for developing IoT research applications. Short and iterative development cycles allow researchers to quickly correct an error in the conception or in their assumptions [11], [12].

Our approach considers that research in LLNs follows the life-cycle as illustrated in Figure 2. A certain number of steps

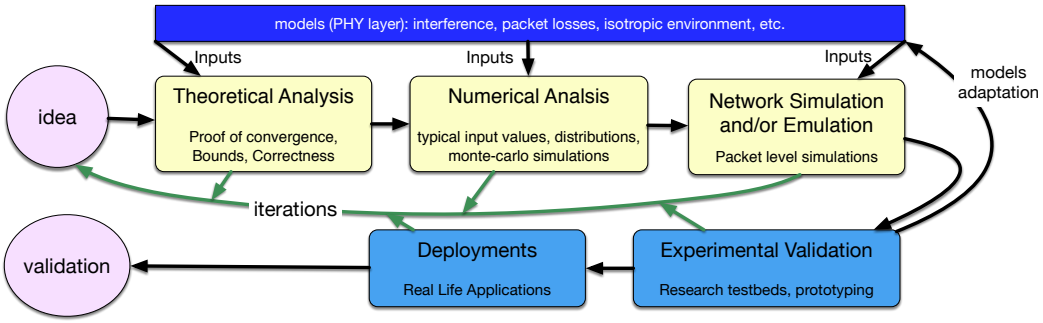


Fig. 2: Typical long-term Validation of a whole solution.

are required to evaluate the performance of a protocol, an algorithm or a whole networking stack. For instance, switching too quickly from the idea to protocol experiments may be prejudicial [105], [14].

A. Theoretical Analysis

We often bootstrap a performance evaluation by analysing theoretically the proposed algorithm. While it is considered to be a preliminary step, it *proves* that the algorithm convergence was correctly designed.

Typically, we may prove the following main properties:

- **Convergence:** An algorithm is designed to get various input and produce specific output. The convergence may not be trivial in some cases, and a formal proof is sometimes required;
- **Self-stabilization:** Whatever the initial state is, the algorithm must converge to a legal state in a finite number of steps [15]. This self-stabilization property is vital for self-healing, while the network is able to correct the transient errors autonomously;
- **Complexity:** The algorithmic complexity is of prime importance for wireless LLNs, since they are composed of embedded, low-cost nodes. The complexity is tightly coupled with the resource allocation (e.g., memory, number of control messages) and hence with the energy consumption;
- **Approximation:** Most of the interesting problems are NP-hard in LLNs, thus requiring to propose heuristic approaches, so that the problem is still tractable by small devices with limited capabilities. In particular, we may prove that an algorithm is an α-approximation, i.e., the algorithm leads to a solution which differs by α from the optimal one [16];
- **Lower and upper bounds:** When proposing an heuristic solution, it is always relevant to provide upper and lower bounds. While the average case might be of interest, focusing on the worst case helps to answer which are the guarantees that could be provided by the heuristic.

B. Numerical Analysis

Obtaining mathematical closed form expressions is practically complicated for most of the problems. A numerical analysis uses different values as input to study practically what the obtained results would be.

With Monte-Carlo simulations, we generate a set of different input values, using a given distribution [17]. For instance, a network should behave similarly, with a different random location of each source node. Monte-Carlo simulations may help to estimate the distribution of the different values (e.g., delay, packet losses) when the models are too complicated to obtain the result analytically. Basically, the characteristics of the sampled random values are close to the real distribution of the values when the number of samples is very large (law of large numbers).

Monte-Carlo simulations may use any mathematical tool to pick random samples. Soua *et al.* use [18] GNU Octave to evaluate the robustness of a scheduling algorithm for a LLN scenario, while Palattella *et al.* [19] use Python scripts and Keshavarzian *et al.* use Matlab [20] for the same purpose.

C. Simulation and/or Emulation

In this subsection, we first describe the tools that simulate a whole network (e.g., energy consumption, radio propagation, protocols). We then will detail the approach to emulate a whole device, while simulating the radio environment only.

1) Network Simulation: Simulation has been widely used for a few decades to validate the behaviour of a protocol. Discrete event simulation is the most popular tool, where a collection of events (packet reception, time-out) are handled chronologically by the simulator. Typically, a pseudorandom number generator is used to mimic uncertainty when probabilities have to be respected (e.g., the Packet Error Rate for a transmission). Thus, simulations help to improve the reproducibility: providing the seed, the simulator version, and the code is sufficient to re-obtain the same results.

For LLNs, the simulations have to integrate PHY models, including the radio propagation. Unfortunately, these radio characteristics are very complicated to capture [21], since they may exhibit very different behaviour depending on many properties (e.g., indoor vs outdoor, ISM band, modulation, urban vs. rural). Similarly, most of the simulators are not hardware specific, and do not model the fine-grained behaviour of each device (i.e., instruction level). For instance, most simulators do not model clock drifts while it may impact very significantly the performance of a network [22].

A large variety of network simulators exists in the literature, either as open source (ns-2 [23], ns-3 [24], OMNeT++ [25]) or proprietary (GloMoSim [26], Riverbed Modeler [27], Qualnet [28]). Their respective interests and limits are detailed in Section VII-B.

Unfortunately, no common API or framework exist for the different simulators. This means that a network stack evaluated on a given simulator cannot be transferred directly to another one. All the protocols have to be rewritten, jeopardizing the reproducibility.

Because reproducibility is now a major concern in the networking community [29], [6], emulation appears as a very relevant solution for a performance evaluation.

2) Emulation: Emulation uses a very fine-grained model of the device to consider, and is consequently hardware specific. Actually, most of the emulators for wireless networks comprise two parts:

1) the node: the emulator mimics the exact instruction-set processing. This way, we can also consider memory and CPU constraints. Energy consumption tends to be more and more accurately modelled if some criteria are respected [30].
2) the PHY layer: most of the emulators *simulate* the radio propagation. We can use here the same PHY models as for simulators.

Thus, the same implementation can be used to execute the code on both the real and the emulated hardware. It greatly reduces the development costs.

The code is often specific to the hardware, at least for the lower layers of the communication stack. Fortunately, there are some very popular platforms which are frequently employed in LLN scenarios (such as the TelosB motes [31]), thus creating a de facto standard for experimenting a set of protocols.

Emulation tools such as Cooja [32] and OpenWSN [33] are now very popular and concentrate most of the developments. By using a limited set of emulators, the research community also encourages code re-use, and reappropriation.

III. Methodology

In this Section, we propose a methodology to evaluate the protocol performance. While this could be considered as a straightforward problem, many research papers still do not present their simulation or experimental results in an accurate and detailed fashion. In this article, we focus on protocols of layers 2 and 3, considering that physical layer is a given parameter of simulations and experiments while upper layers strongly depend on the target application. In our humble opinion, MAC and routing protocols thus, constitute the backbone of the communication stack of LLN devices. Consequently, hereafter, we present the different metrics and guidelines that appear specific to the thorough study of these two aforementioned layers. We also highlight the importance of conducting reproducible, robust and consistent performance evaluations.

A. What should we measure?

The first thing a researcher should wonder when evaluating a protocol / algorithm concerns the target metrics to be measured (cf. Table I). Hereafter, we list some common criteria for each network layer as follows.

1) Common metrics: Some metrics are not relative to a particular communication layer and should always be monitored in wireless LLN scenarios.

a) Energy consumption: First of all, a LLN aims at integrating low-power nodes using batteries. Thus, the radio chipset has to be turned off to save energy [34].

We may measure the average duty cycle ratio that represents a first approximation of the network energy consumption. More precisely:

$$DCR = \frac{T_{active}}{T} \quad (1)$$

with T being the considered time-interval and T_{active} the time duration which the radio chipset of the node is set to active. The popular CC2420 chipset typically consumes 18.8mA in RX mode, between 8.5 and 17.4mA in TX mode, 0.4mA in idle mode, and $0.02\mu A$ when sleeping. If we neglect the difference between the idle, TX and RX modes, the duty cycle ratio can be a good representation of the energy consumption of a network node.

b) Network Lifetime: We may measure the network lifetime, which could be defined as [35]:

- **First death:** When the first node runs out of energy;
- **Last death:** When the last node dies [36]. However, such definition assumes implicitly a single hop topology, with all the sensors being equivalent (i.e., with no heterogeneity);
- **Connectivity:** The size of the largest connected component exceeds a threshold value. However, this metric is extremely dependent on the application, since it assumes that the sensors are redundant, which may not hold in practice;
- **Coverage and Connectivity:** A collection of regions have to be monitored and, thus, a sensor is able to measure a given phenomenon in a given geographic area, i.e., it actually *covers* this area. Each area must be *covered* by at least k sensors and these sensors are able to send their measurements to the central sink (i.e., connectivity).

The energy consumption and the network lifetime are closely related metrics. However, most of the researchers focus on the primer because it is more convenient to measure, while on the other hand, measuring accurately its lifetime, the network has to operate and get monitored for very long durations.

c) Overhead: A chatty protocol needs to exchange many packets and consequently it consumes both bandwidth and energy. Therefore, it is important to measure the network overhead, expressed as the quantity of control packets generated by all the nodes.

This overhead may be measured in:

- packets per second, when the cost to access the medium is high (e.g., long preamble sampling, the data packet length being small compared with the sum of the preamble and the data frame);
- bits per second, when the transmitter and the receivers have to stay awake only during the packet's transmission;

Layer	Metric	Section	Definition
Transversal	Energy Consumption	III-A1a	Quantity of energy consumed by all the nodes
	Network Lifetime	III-A1b	Time before the network stops doing its job
	Overhead	III-A1c	Amount of control packets (bits or packets per second)
MAC	Delay	III-A2a	Medium access delay, comprising also the retransmissions
	Packet Delivery Ratio	III-A2b	Ratio of packets correctly received by the receivers
	Fairness	III-A2c	Jain Index
	Duplicates	III-A2d	Number of packets duplicated (i.e. forwarded in multiple exemplars) by the MAC layer
Routing	Delay	III-A3a	End-to-end delay, considering both medium access and buffering delays
	Packet Delivery Ratio	III-A3b	End-to-end packet delivery ratio from the source to the final destination

TABLE I: Definition of the different metrics.

Some researchers have proposed to divide the number of control packets by the number of packets received by the sink in order to measure network *efficiency* [37]. We consider that this corresponds to two different criteria that should be measured separately.

2) Medium Access Control (MAC): Medium access represents a key challenge in wireless LLNs. MAC layer is in charge of determining when nodes should transmit, receive, listen or simply remain idle. It thus handles all operations related to the main source of energy consumption, which is packet transmission and reception [38]. In particular, a node must turn its radio off most of the time because it is the only way to save a significant amount of energy in radio networks [34]. The lowest this *duty cycle ratio* is, the larger the network lifetime will eventually be.

Similarly, the MAC protocol must solve the contention amongst different interfering transmitters. This approach often provides a trade-off between reducing the medium access delay, while increasing the collision probability, and conversely.

In the literature, several MAC strategies have been proposed for LLNs [39] (Figure 3):

- **Preamble sampling:** The transmitter sends a long preamble, and the receiver has to turn its radio on periodically to detect this preamble. Then, the receiver has to stay awake in order to receive the data packet, which follows the preamble [40].
  Contiki-MAC [41] represents a widely used preamble sampling protocol, which exploits several optimizations for reducing the energy consumption.
- **Synchronous protocols:** The receiver and the transmitter must agree on a common schedule to wake-up synchronously so as to be able to exchange frames. The transmitter maintains a guard-time before its transmissions for compensating possible clock errors [42], [43]. In its Time-Synchronized Channel Hopping (TSCH) mode, IEEE 802.15.4 [44] uses a Time Division Multiple Access (TDMA) matrix paired with slow channel hopping to combat external interference;
- **Wake-up radio:** Each node must have two interfaces. The (always on) wake-up radio is in charge of waking-up the receiver [45]. The second radio provides higher bit rates and exchanges data packets.

a) Medium Access Delay: Medium access delay corresponds to the time separating the frame dequeued from the buffer and its reception by the next hop. However, we need to clarify that we do not refer to the buffering delay, since it corresponds rather to a problem concerning higher layers.

We may consider the transmission successful as soon as the data frame is received by the next hop. Unfortunately, the transmitter is not certain if its transmission was successful, because the communication channel is half-duplex and the radio link may be unreliable. Thus, the latency is often approximated by the time difference between the data generation and the reception of the corresponding received acknowledgement.

If we consider broadcast packets, the problem is more complicated since we have to consider the following aspects:

- **Time distribution:** A broadcast packet may not be received at the same time since some MAC layers implement a duty cycle approach and, thus, a copy of the same packet may be received by neighbours according to their sampling frequency;
- **Reliability:** Only some neighbours may have received the broadcast packet. In this case, for instance, we could compute the time at which 75% of the neighbours have correctly received the broadcast packet.

b) Reliability: Packets may be lost because of collisions or link unreliability. Thus, it is mandatory to measure the Packet Delivery Ratio (PDR) as:

$$PDR = \frac{N_{rx}}{N_{tx}} \tag{2}$$

with N_{rx} the number of received packets, and N_{tx} the number of transmitted packets. This approach is considered as a layer 2 metric; if a packet is received after two consecutive retransmissions, the PDR is considered equal to $\frac{1}{3}$. Since these retransmissions are hidden to upper layers, measuring the PDR at layer 2 is of primary importance. More retransmissions also mean a higher energy and bandwidth consumption.

The throughput consists in the PDR of a given flow multiplied by the rate of the corresponding flow. Since it does not reflect the actual *pressure* on the network (e.g., some packets may be dropped at intermediary nodes), this metric is less relevant for LLNs.

It is also important to distinguish the packet losses between those that are caused by collisions, and those that emerged from low quality radio links. In simulations, we can measure accurately the number of packets that overlap at some receivers (i.e., the collisions). In particular, the centralized scheduler

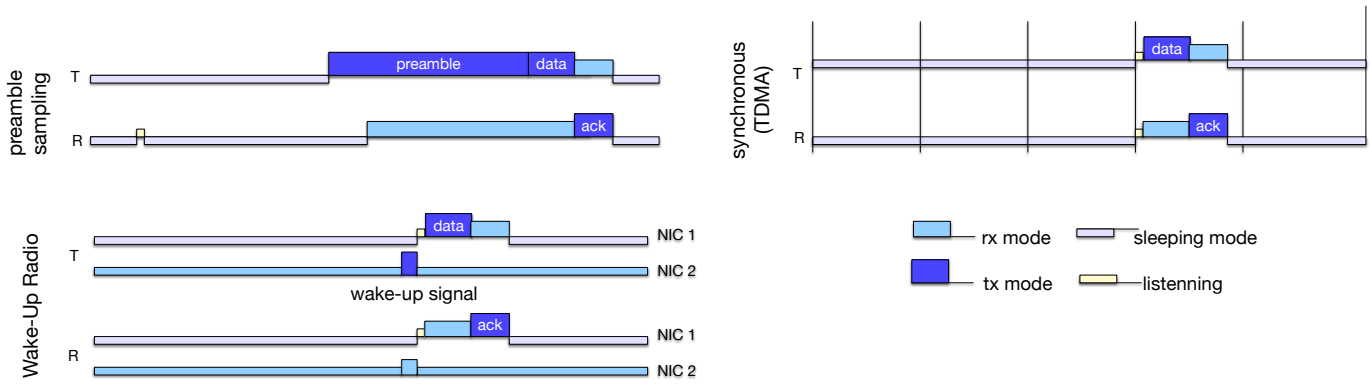


Fig. 3: MAC approaches for low-power and lossy networks.

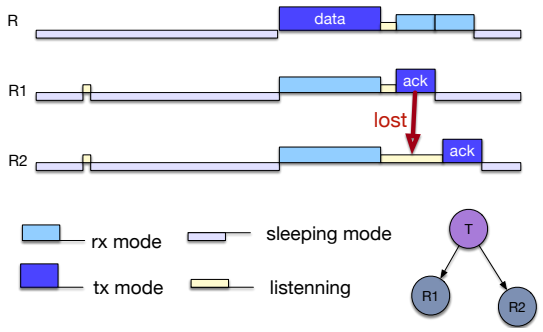


Fig. 4: General of duplicates with an opportunistic MAC.

computes the signal strength of all frames in every receiver. Unfortunately, isolating collisions in experiments is a much more complicated task. Giustiniano *et al.* [46] proposed to classify the source of packet loss for IEEE 802.11 networks by measuring the loss. However, this technique has to be adapted to each considered MAC layer.

Finally, we may also isolate the source of external interference that results in packet loss. Techniques such as SONIC [47] classify the source of interference (e.g., from WiFi or Bluetooth networks) based on the variation of the signal strength and payload corruption.

c) Fairness: Especially in wireless LLNs, we have to guarantee a fair behaviour, which means that each transmitter should receive the same amount of radio resources. We often utilize the Jain Index [48] in order to measure fairness as follows:

$$JainIndex(x_1, .., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n * \sum_{i=1}^{n} x_i^2} \quad (3)$$

where x_i is the considered performance metric (i.e. PDR, delay or throughput) achieved by the flow / source i.

A Jain Index equal to 1 means a perfectly fair solution. The lower is the Jain Index, the more unfair the solution behaves.

d) Duplicates: Since transmissions are unreliable, data packets may be correctly received and the corresponding acknowledgement may be lost (due to an unreliable link) or even may collide (due to the hidden terminal problem [66]). Asymmetrical links [50] tend to exacerbate the loss probability of this acknowledgement packet. When the transmitter does not receive the acknowledgement, it schedules a retransmission, leading to a duplicated packet. Mechanisms such as sequence numbers help to limit the impact of such problems [51].

Furthermore, opportunistic forwarding in the MAC layer exploits the broadcast nature of the radio transmissions in order to alleviate unreliability problems [52]. A single packet is transmitted using the anycast mode to a list of (ordered) next hops (see Figure 4). $R1$ correctly receives the packet and acknowledges it. The `ack` is received by T but not by $R2$: the second receiver will also consequently send an `ack` and will enqueue the packet. Since in this case duplicates are generated in different nodes ($R1$ and $R2$), their detection is more challenging to be achieved.

We usually measure the duplicate ratio (ratio of the number of packets received by the destination and the number of packets really generated by the source). We consider the number of end-to-end duplicates because it captures both the negative (higher overhead) and the positive impacts (higher reliability).

3) Routing: Routing aims at deciding which node should be used to relay the packets toward a given destination [53]. The main following approaches exist in the literature:

- **Proactive:** Routes are constructed a priori toward the destination, thus, all nodes in the network are aware of the routes to any destination at any time. Therefore, since all routes are stored in the routing tables, a node may transmit its data packets to the destination at very low delay. To maintain the routing tables constantly updated, periodic routing-related control packets are required to be transmitted.
  The Routing Protocol for LLNs (RPL) represents the standardized proactive routing protocol for LLNs, specified by the IETF ROLL WG [54]. It is a distance vector routing protocol and it is defined as link-layer agnostic. Thus, RPL may operate over wireless or PLC networks. RPL builds a Destination Oriented Directed Acyclic Graph (DODAG) whereby each node chooses one or more parent(s). Each node compute its *rank* representing its *virtual* distance from the root. More precisely, the node uses an *Objective Function* (OF) to derive its rank from the rank and the link quality of its best neighbour (its parent);

- **Reactive:** Routes are built on-demand, when the source has packets to transmit, which means that it is not necessary to maintain a route if there is no traffic. Thus, reactive protocols do not require routing-related information to be transmitted periodically. It relies on a flooding mechanism to reach the destination, and to construct the routes.
  Lightweight On-demand Ad Hoc Distance-vector Routing Protocol Next Generation (LOADng) [55] is reactive routing protocol based on Ad Hoc On-Demande Distance Vector Protocol (AODV). At its core, it floods the network with a route request, and the destination unicasts a route reply after having inserted the route in its routing table;
- **Geographic:** If all the nodes are aware of their geographic location, they may select greedily the next hop as the neighbour closest to the destination. However, such local rule leads sometimes to a cul-de-sac, not easy to tackle [56].

A low-power lossy network may comprise multiple sinks, connected to the rest of the Internet [57]. These different sinks may be used for instance to balance the load, or to increase the available bandwidth for each individual device. The sinks may also be used to improve the robustness, each device having several independent routes to the Internet (through each different sink).

When considering a multihop network, we aim at focusing on the end-to-end performance. To simplify this interpretation, we often only consider the performance in the multihop wireless part.

a) L3 End-To-End Delay: The layer 3 (routing) delay corresponds to the multihop delay taking into account the following:

- **Medium access:** The MAC delay (contention, possible retransmissions, and reception of the acknowledgement);
- **Queueing:** A packet is received by a node and stored in a temporary queue before being forwarded.

In particular, a routing protocol, which efficiently balances the load may help to reduce queuing delay. In the same way, different queuing strategies (e.g. FIFO, LIFO, Priority Queuing) may impact the distribution of the end-to-end delay.

b) End-to-End reliability: The end-to-end PDR counts the ratio of the number of packets received by the destinations and transmitted by the sources. We often neglect the incurred loss outside the wireless part.

High reliability is a requirement for many industrial LLN applications [58]. Thus, achieving an almost perfect PDR (close to 99.99%) may constitute an important requirement that needs to be satisfied.

Since there is no need to receive all the measures when they are redundant, reliability may also be application specific. For instance, we consider that an intrusion detection system fulfills its objective if the event was detected by the system, whatever PDR is obtained. We then propose to measure both the Over and Under Detection Ratios as follows:

$$OverDetectRatio = \frac{N_{falsePos}}{N_{Pos}} \quad (4)$$

$$UnderDetectRatio = \frac{N_{falseNeg}}{N_{Pos}} \quad (5)$$

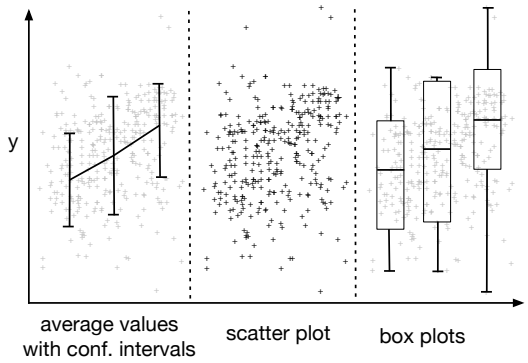


Fig. 5: Representation of experimental data: average values, scatter plot, boxplots.

where N_{Pos} is the number of times that the system detected a phenomenon (e.g. an intrusion, a location with a pollution higher than a threshold, etc.), $N_{falsePos}$ is the number of false positive detections (detected erroneously as true), and $N_{falseNe}$ is the number of false negative detections (not detected at all).

We often target to achieve $UnderDetectRatio \approx 0$ because we cannot accept to *miss* detecting an event, while minimizing $OverDetectRatio$ since it triggers useless actions (e.g. network reconfiguration or human intervention).

B. How should we measure?

We here provide some general guidelines about measuring the aforementioned metrics.

1) Average value: In order to obtain robust results, we have to verify that they are *representative*. Thus, we often run a simulation / experiment several times by utilizing the same input. We then present the average value $\bar{x}$ for all the experiments / simulations with the same input values. Alternatively, the median value is less sensitive to outliers and provides an accurate manner to synthesize a set of different experiments.

However, we have also to appreciate the distribution of the different values, particularly when we compare different solutions. The confidence interval helps to quantify the variability and to verify that a difference is significant. Let us assume that the values follow a normal distribution. The typical 95% confidence interval is defined as:

$$\left[\bar{x} - 1.96\frac{\sigma(X)}{\sqrt{|\mathcal{E}|}}, \bar{x} + 1.96\frac{\sigma(X)}{\sqrt{|\mathcal{E}|}}\right] \quad (6)$$

where $\bar{x}$ represents the average value, $\sigma(X)$ is the standard deviation of the random variable and $|\mathcal{E}|$ is the number of experiments carried out. A 95% confidence interval means that if we pick randomly an experiment, it has 95% probability to perform within the computed interval (see Figure 5).

2) Distribution and worst case: While the average value provides to us an overview of the *average* case, we should also consider the best and worst cases.

Boxplots are a convenient way to plot the distribution of the values as well as the minimum and maximum values. We

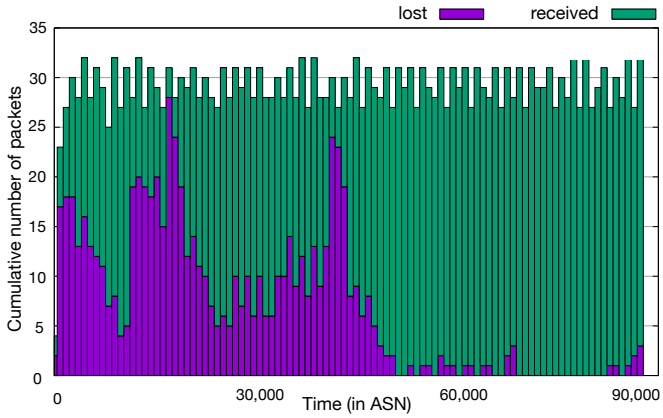


Fig. 6: Convergence of a distributed scheduling algorithm in a 6TiSCH stack [60].

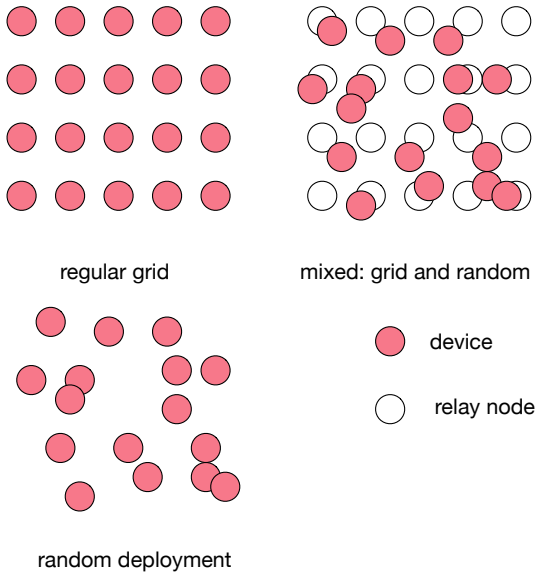


Fig. 7: Common topologies for performance evaluation.

may also use a beanplot [59] to compare two protocols, two by two. Thus, we are able to quantify visually the difference in the distribution. This visual representation is complementary to the Jain Index presented in Section III-A2c.

Let us now consider Figure 5 in which one boxplot represents the measurements for a given x coordinate interval. Each box represents the median value, the first and third quartiles for all these measurements. The whiskers here represent the minimum and maximum values. We are able to detect tendencies and to estimate more accurately the scattering.

Particular attention has to be given to the outliers, i.e. measures distant from other observations. They may mean:

- **Measurement error:** The metric was erroneously measured (e.g. due to inaccuracy in the sensor device);
- **Bug:** The protocol was erroneously implemented;
- **Algorithm:** The algorithm was badly designed and it does not converge under certain situations;
- **Heavy-tailed behaviour:** The values do not follow a normal distribution and *certain* situations lead to a very bad case.

Depending on the situation, researchers will have to identify the exact cause(s) and if possible to patch the solution.

3) Initialization vs. steady-state: We make a distinction between the following two steps:

- **Initialization:** The node just booted and has not joined the network since it has to acquire an address (e.g. local, global), the cryptographic keys (security), a route (routing protocol) or a parent (MAC and topology control).
- **Steady-state:** The node has finished its configuration and can now send and receive packets.

Let us consider Figure 6 that illustrates the PDR of a 6TiSCH stack [60], with a distributed scheduling approach, during 30 minutes. The bars represent the volume of packets generated during a time window. The violet (bottom) part depicts the packets that are never received by the sink. We can identify the initialization period (before the slot $50,000$) and the steady-state period (i.e. the algorithm has converged).

4) Time Dependency and Stability: Surprisingly, most protocols and algorithms are evaluated under stable conditions. For instance, the simulation model is assumed to remain unchanged for the whole simulation period. Unfortunately, node failure [61] or link quality changes [62] are practically very common.

A routing protocol may even not converge under stable topologies when stochastic metrics are used [63]. Some statistical estimators such as a Kalman Filter have often to be used [64].

Thus, these time variations have to be carefully studied during a long time period.

IV. Radio Topology

Many papers evaluate their proposition in a random topology, using a synthetic model to mimic the radio characteristics. We propose here a comprehensive view of the possible key scenarios and their respective limits. We make a distinction between:

- **Network topology:** It provides a high-level view of the set of motes which can directly communicate with each other. Too many papers focus uniquely on a random topology, which complicates the interpretation and which may *hide* the pathological situations;
- **Communication model:** It describes formally the rules which enable a pair of nodes to communicate with each other. We often associate the Bit Error Rate (BER) metric to each radio link; it denotes the probability that a given bit in a frame is erroneously received;
- **Interference model:** If two transmissions start simultaneously and the receivers are close from both transmitters, the signals will probably collide, wasting the radio bandwidth. The interference model defines formally in which case a transmission undergoes a collision.

A. Network Topologies

The topology strongly impacts the performance of the network. We make a distinction between the following devices:

A sensor measures a physical phenomenon (e.g. temperature, wind) and sends these measurements to a processing entity;

An actuator has a physical action on the environment (e.g. open a door/window, control of HVAC);

A relay node just forwards the packet via a low-rate radio interface;

A sink (or border router, or gateway) connects the low power wireless network to the Internet and is often considered plugged in.

All these devices are not involved similarly in the traffic process (see section V). For instance, an actuator *consumes* most of the time a command while a sensor *generates* a measure.

Some scenarios have been identified in the literature to create some specific problems. Thus, they should be studied in isolation, to verify that the protocols and algorithms work accurately in these situations. We distinguish the scenarios common to any protocol for LLNs, and those specific to a particular layer, where a particular problem may arise.

1) Common metrics: Some scenarios are layer-agnostic, and should be studied regardless of the protocol.

a) Regular grid: The easiest large scale topology to set-up consists of a regular 2D or 3D grid (Figure 7). A node is a radio neighbour with the closest nodes in the grid (with an almost perfect link quality). This ensures that only good links are used to forward the packets.

b) Mixed Grid: Alternatively, the regular grid would consist of the relay nodes (i.e. they don't generate traffic but just forward the packets of others) and the devices (sensors and actuators) are placed randomly in the area. In this way, we guarantee a network-wide connectivity, while also considering a heterogeneous deployment of the *clients*.

c) Random: The random topologies are the most common scenario; a fixed number of devices is randomly placed in the considered area. If we distribute a sufficient number of nodes, the topology is connected with a high probability [65].

2) MAC layer:

a) Star topology: The simplest topology is the star topology in which all the devices send their packets to a central node (Figure 9). We are able to evaluate the performance of the MAC layer under the most basic scenarios, where all the devices hear each other and the radio links are perfect (due to the fact that all devices are sufficiently close to the central node).

Throughput and fairness performance should be optimal in this case.

b) Hidden terminal: The hidden terminal problem is known for a long time in radio networks [66]. When two transmitters do not hear each other and send their packets to the same receiver, collisions are very frequent. A modern MAC layer should address this problem.

Isolating this particular situation helps to focus on the hidden terminal problem, proving that the protocol deals efficiently with this unwanted situation.

c) Line: A line represents the atomic pattern of a multi-hop network; packets are forwarded along a path from a source to its destination. Unfortunately, this simple topology has been proved to perform poorly with random access [67].

For medium access, we should evaluate the behavior of the MAC protocol in the following situations:

- Unidirectional: One of the extremities send frames to the other extremities. This models well a sensor that sends its measurements in the cloud via the sink;
- Bidirectional: Both extremities exchange frames with each other. Indeed, communication with actuators or mote's reconfiguration require a bidirectional exchange to make reliable the transaction.

d) Heterogeneous radio link qualities: The evaluation should also consider heterogeneous radio link quality. In Figure 9, we consider a star where two radio links present a large Packet Error Rate (PER); a frame has a non negligible probability to not be received correctly.

The MAC protocol may adopt one of the following approaches:

- Transmission opportunity fairness: Each transmitter has the same probability to access the medium. The receiver with the poorest radio link quality will have the lowest bit rate;
- Bandwidth fairness: Each receiver has the same *goodput* (bit rate of frames correctly received).

3) Routing protocol: At the routing layer, we should consider the fairness among different flows, and its ability to support load-balancing.

a) Ladder (path quality): The simplest scenario consists of a ladder in which the source sends packets to the sink and several redundant paths exist. All the radio links experience different quality (i.e. PER). Thus, the different paths provide different end-to-end reliability and latency.

We can consequently evaluate the ability of the routing protocol to exploit the most efficient route.

b) Ladder bi-source (load-balancing): An efficient routing protocol should be able to balance the network load for congestion avoidance. If the objective corresponds to maximizing the network's lifetime (see Section III-A1b), we must balance the energy consumption. More precisely, all the nodes should consume an equivalent quantity of energy, while minimizing the maximum energy consumption [68].

With a ladder of two sources, we can validate the ability of a routing protocol to optimize the load/energy-balancing property under the simplest scenario.

c) Synchronous vs. Gradual Bootstrapping: In a *synchronous* scenario, all the nodes boot simultaneously and start to execute the algorithm / protocol. While most of the performance evaluation focus on this particular scenario, its practical interest is limited. Indeed, it assumes that all the nodes are first deployed and then the operator commands remotely their start-up. The trigger of this start-up is never described, while it actually represents a key challenge. Indeed, even the flooding of a `startup` control packet may be insufficient since a blind flooding is expensive and unreliable [69].

It seems that the *gradual* bootstrapping is far more realistic. The sinks are first deployed and then new devices are gradually inserted in the network. This scenario models a network where

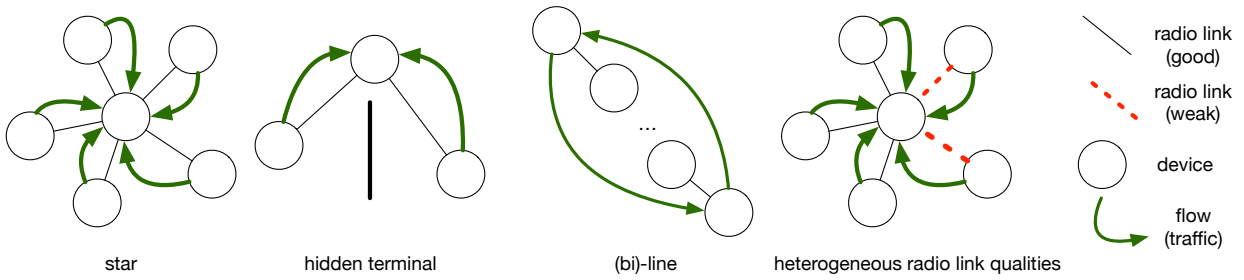


Fig. 8: Benchmarking topologies for the MAC layer.

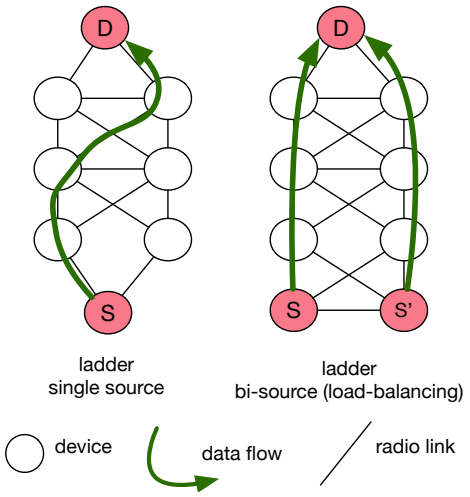


Fig. 9: Benchmarking topologies for the routing layer.

new applications / nodes / services are iteratively deployed. For instance, smart meters may be progressively deployed in a given district [70]. Consequently, they start to generate periodical measures as soon as they are installed (or replaced) by the subscriber.

To avoid wasting the energy resource of the nodes, the new devices should be able to join the network as soon as they start. Thus, we may assume a gradual deployment with connectivity preservation; a neighbour with a sufficient link quality already exists in the vicinity of the new node.

Surprisingly, this *gradual bootstrapping* scenario is very seldom studied. Most of the experiments focus rather on an ideal case, where all the nodes wake-up and have to join the network simultaneously.

B. Communication model

Simulation and theoretical analysis rely on a PHY layer model and they try to mimic radio propagation. Such a model often associates with each pair of nodes a success probability (i.e. a packet would be received correctly or not). In a asymmetrical link, this probability is different for the two directions;

1) Unit Disk Graph (UDG) communication model: The simplest model consists of a geometric communication model in which two nodes can exchange packets only if they are located at most 1 unit distance far apart. Furthermore, the radio link quality is assumed to be perfect for any radio link with a length inferior or equal to 1 unit.

Let $G(V, E)$ be a graph G with the set of vertices V and edges E. A UDG is defined by:

$$\forall (u, v) \in E, d(u, v) \leq 1 \quad (7)$$

$$\forall (u, v) \notin E, d(u, v) > 1 \quad (8)$$

where $d(u, v)$ denotes the euclidean (geographical) distance from u to v.

A variant of the UDG consists of using a fixed radio range ($\neq 1$). However, it just consists of rescaling the geographical distance.

Comments: the UDG model is very simple and is still used for this purpose. However, it has been proved to largely overestimate the performance achievable in realistic conditions. Since they do not consider at all the **unreliability problem** (radio link quality), we consider that the UDG model is inappropriate, and **should not be used** for a performance evaluation in multihop wireless networks.

2) Free space: The Free Space Path Loss (FSPL) model estimates the attenuation of the signal when no obstacle is present between the receiver and the transmitter (line-of-sight scenario). Let P_t (resp. P_r) be the transmission power (resp the reception power). We have:

$$P_r = G_t(u) G_r(v) P_t * \left(\frac{\lambda}{4\pi d(u, v)} \right)^2 \quad (9)$$

with λ denotes the wavelength ($\frac{f}{c}$), $G_t(u)$ and $G_r(v)$ respectively the transmission and reception gain of u and v.

We can compute with this path loss the strength of the received signals at the receivers and we can deduce the Signal to Interference plus Noise Ratio (SINR). The SINR and the modulation then define the BER which corresponds to the probability that a bit is received in error.

Comments: This free space model considers an ideal scenario, without obstacle. In conclusion, this model should not be used for a performance evaluation since more sophisticated models provide a better realism without decreasing the tractability.

3) Two ray ground: The two ray ground model was widely popularized by the NS2 simulator [1] and it considers both the direct and ground reflection paths. When the transmitter and receivers are close to each other, the path loss is defined by

[1] http://www.isi.edu/nsnam/ns

Equation 9 and the impact of the reflected wave is considered negligible. In all cases, the received signal strength is denoted by:

$$P_r = G_t(u)G_r(v)P_t * \frac{h(u)h(v)}{d(u,v)^4} \quad (10)$$

where $h(u)$ is the height of the antenna of node u.

Comments: This model reflects more accurately the gray zone in which the radio link is neither short nor long and the radio link quality significantly varies even for a small change of distance. However, it keeps on considering only ideal environments, e.g. omnidirectional antennas, isotropic and homogeneous environments. In particular, this model does not capture the time variability which makes the radio link so difficult to exploit [71].

4) Log-normal path loss model: The Log-normal path loss model tries to empirically estimate signal strength. It accurately estimates the path loss inside a building or in a high-dense populated area (e.g. a smart city). The path loss in Decibels is defined by:

$$PL_0 + 10\gamma log_{10}\frac{d(u,v)}{d_0} + X_g \quad (11)$$

where PL_0 is the path loss measured at a distance of d_0, and X_g is a normal gaussian variable with zero mean modeling the flat fading.

This constitutes, to our mind the most accurate model for Wireless Sensor Networks. It is sufficiently **generic** to mimic a common situation, while still considering heterogeneous and unreliable radio links. More specific models, tailored for a given scenario (forest, indoor, etc.) can rather be addressed using a genuine dataset (see section IV-E).

5) Experimental measures to calibrate the models: The parameters values strongly impact the characteristics of the radio topology. Testbeds may have this purpose: measures in realistic conditions may help to find the most accurate parameters values.

For instance, Chen *et al.* proved that a Log-normal path loss model is realistic to mimic indoor situations [72]. They identified several scenarios for which they calibrated the models appropriately.

Vazquez *et al.* [73] considered rather a smart city scenario, using the 868MHz band, and estimated the path loss in different situations, depending on the location of the antenna (on top of a building vs. in the street).

C. Interference Model

If two transmitters simultaneously send a frame, both radio signals may interfere at one of the receivers. This collision consumes energy and bandwidth and, thus, should be avoided. The interference models aim to reference the list of mutually interfering radio links.

We have to carefully select the right interference model, since it strongly impacts performance, whatever the stack of protocols is [74]. We detail the most common models in Figure 10.

1) Fixed range: Similarly to the UDG, see Section IV-B1, we assume a fixed interference range. Any node that is located in the interference area is not allowed to transmit or receive any packet. This area is modeled by a disk, centered in the transmitter, with a radius equal to the interference range (typically twice the radio range).

Comments: This fixed range cannot model the capture effect, where a signal can be decoded correctly only if the transmitter is very close to the receiver. Furthermore, it also assumes a circular interfering region, which leads to an homogeneous network topology, and tends to simplify the task of the network algorithms. This model **should not** be considered for a realistic performance evaluation.

2) k-hops interference model: This graph based model expresses the interference rule as a hop distance condition. Basically, a given transmission fails if another transmitter, located less than k hops away from the transmitter or the receiver is active at the same instant. Such model assumes implicitly that the transmission (and interference) is bidirectional (the data and acknowledgement frames).

Let consider the topology illustrated in Figure 10-b (1-hop interference model). All the nodes neighbours of S and D cannot neither transmit nor receive any packet.

Comments: This interference model is often used in the algorithms executed to e.g. allocate resource. However, the model does not capture the radio effect, nor heterogeneous situations. Another more realistic interference model should be preferred.

3) SINR: Since radio signals are additive, two pairs may not collide but inserting a third one may create collisions. The SINR estimates the radio signal strength at the receiver. Depending on the modulation scheme, this level is then translated into a BER.

Any of the previous radio propagation models may be used to add up the signals received from the interfering nodes and the ones from the transmitter: we have to compute the ratio of the signal strength of the transmitter and those of the interfering nodes at the receiver. Then, the radio chipset specifications (with the modulation, the bitrate, etc.) provide the corresponding Bit Error Rate (BER).

Comments: The SINR model represents the most generic model, able to capture realistic situations when it is coupled with an accurate radio propagation model. This model is the most realistic one for simulations or a theoretical analysis.

4) Graph of conflict: The conflict graph is a very convenient tool to identify colliding situations. Each radio link consists of a vertex in the conflict graph and an edge exists between the two vertices if the pair of radio links mutually interferes. With a k-hop interference model, it corresponds to the transitive closure of the line graph of the communication graph. The radio links that are authorized to simultaneously transmit without colliding form a Maximum Independent Set in the conflict graph (i.e. they are not neighbours of the conflict graph), (e.g. the radio links SD and BU1 in Figure 10). However, this model is still unable to take into account the capture effect

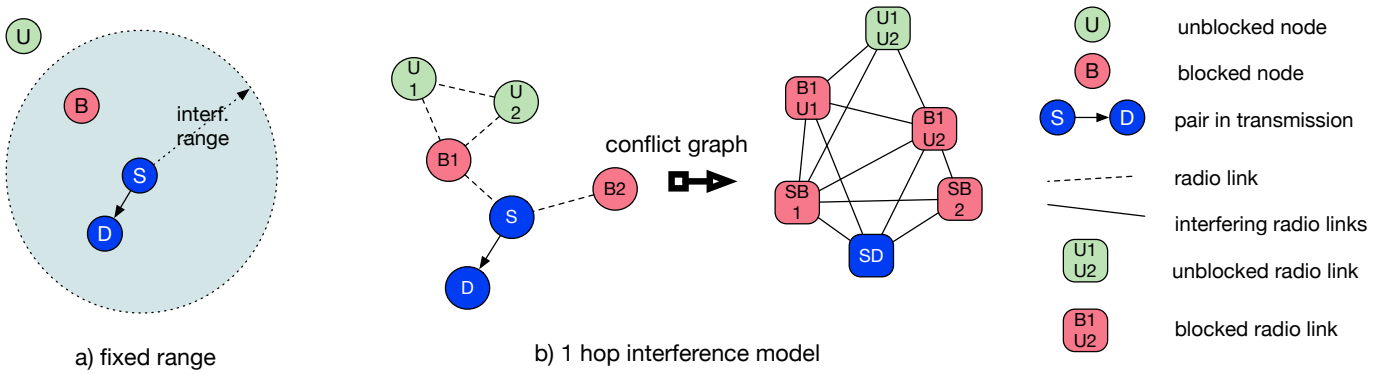


Fig. 10: Different interference models.

D. Key realistic characteristics

The characteristics of multihop wireless networks have been extensively studied in the last years. The different experimental studies have highlighted the existence of some key characteristics which are often not captured by the different existing synthetical models.

1) Time Variability: A multihop wireless network exhibits in most cases a very large time variability. Srinivasan *et al.* demonstrated the existence of a very short term radio link consistency [75]. Basically, some links exhibit a strong burstiness: packet losses are not independent, and the burstiness metric denotes this conditional probability of correct reception.

Cerpa *et al.* studied long and short term temporal aspects [76]. They advocate the obligation for protocols to consider the variations, and to not consider uniquely the *average* case. Because a radio link may drop *all* the packets during a given duration, a huge number of retransmissions may be required during these *bad* periods.

To the best of our knowledge, no model succeeds to capture such variability.

2) Asymmetry: Radio link asymmetry is frequent in LLNs [50]; the Packet Reception Rate is different in both directions. This asymmetry may come from different antennas, transmission power, or the presence of external interference. Radio irregularity impacts MAC and routing layers, but also the accuracy for localization, or the coverage [77].

E. Datasets

Instead of defining synthetic models, probably unable to capture all this complexity, we can also directly inject measures in the simulation / emulation. Typically, a set of packets is generated to test all the radio links, to record their quality, and the occurence of collisions.

Concretely, we have to test all the radio links. Let n denote the number of nodes. Since signals are additive, we have theoretically to test all the possible subsets of radio links to estimate the amount of collisions:

$$n * (n-1) * (n-2)! \tag{12}$$

which becomes quickly intractable. Padhye *et al.* proposed some heuristics to reduce the number of probes [78].

Moreover, these probes have to be achieved continuously for a very long time (i.e. days or weeks). The simulator will then pick-up the experimental result which is the closest from the simulation time.

Comments: this method is very powerful to mimic realistic conditions, while still providing reproducibility. It can accurately model the capture effect, the time variability, etc. However, a dataset if specific to a given scenario, and its exhaustivity is very complicated to guarantee. Public datasets such as soda[2] or tutornet[3] (buildings), hydrobionets[4] (industrial plant), or intel[5] (lab) may be used.

V. Traffic Characterization

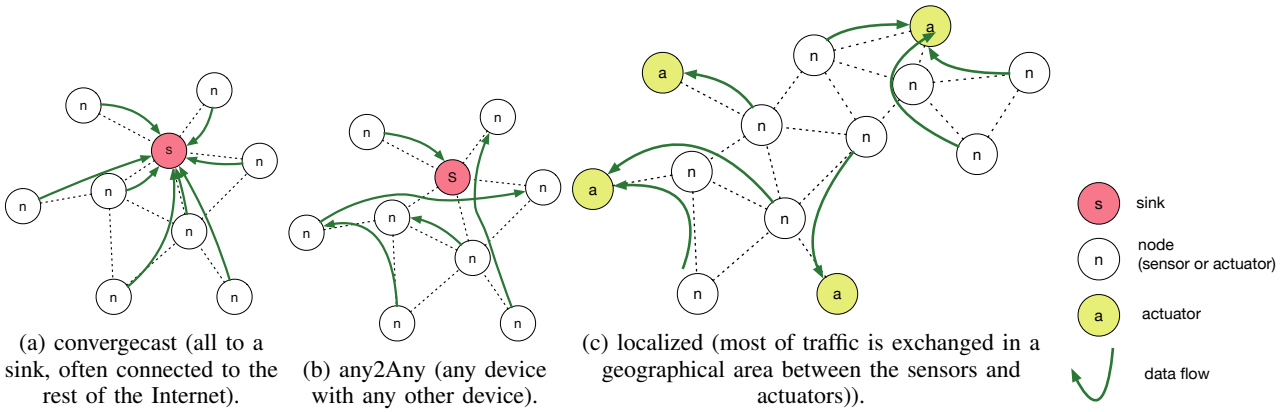
As previously presented, LLNs support a large variety of real-world use-cases, ranging from industrial to monitoring applications. In this Section, we detail all possible traffic patterns and profiles that LLN deployment may support.

Hereafter, we make a distinction between the following different types of traffic:

- **Flooding:** A packet is generated by a node or the sink and has to be delivered to all the nodes in the wireless network. Blind flooding has been proved to perform poorly in wireless multihop networks, leading to the so-called *broadcast storm problem* [79]. Protocols such as Gossiping [80] have been proposed to improve the reliability while reducing the incurred overhead.
- **Multicast:** A packet has to be delivered to a group of motes. Directed Diffusion [81] was one of the first protocols to implement natively multicast: a node floods its interest (a kind of multicast address). When a packet is generated, it is forwarded along the gradient of interests and, thus, delivered natively to all the interested nodes.
- **Unicast:** The most common scenario consists of generating packets with a specific unicast address. Alternatively, a mote may specify an anycast address, e.g., *any* sink connected to Internet.

During the previous decade, in most of the deployments, the employed protocol addresses only *one* of the previously

[2] http://wsn.eecs.berkeley.edu/connectivity/
[3] http://anrg.usc.edu/www/tutornet/
[4] https://github.com/apanouso/wsn-indfeat-dataset
[5] http://db.csail.mit.edu/labdata/labdata.html



(a) convergecast (all to a sink, often connected to the rest of the Internet).

(b) any2Any (any device with any other device).

(c) localized (most of traffic is exchanged in a geographical area between the sensors and actuators)).

Fig. 11: Model of traffic in low-power lossy networks, i.e. type of communications privileged in the topology.

mentioned scenarios. However, since the requirements for each of them are often too heterogeneous to be fulfilled by a single solution, the current standardization bodies and research lead to design and developing standards that may provide solutions for multiple applications over single deployments [82]. For instance, having different traffic flows (i.e., tracks) for different applications [60].

A. Traffic pattern

1) Convergecast: Most common use-cases comprise a low-power devices connected to a cloud infrastructure. The sensor nodes transmit their data packets to the cloud through the sink. The term *convergecast* is used, since all multihop traffic traverse through the border-router (Figure 11a).

Under convergecast traffic pattern, the heavily loaded area is located around the boarder-router, forming the so-called funnelling effect [83]. Thus, all traffic is forwarded by the 1-hop away from the boarder-router nodes, leading to the formation of a congested zone [84]. Therefore, optimizing the traffic pattern in the interfering zone is essential to improve the network capacity [85].

The research community has been working on MAC and routing protocols to particularly focus on addressing this specific challenge. Indeed, the standardization community is progressing to design an efficient time-slotted and frequency-hopping MAC standard [44].

2) Up and Download: The download direction is often not considered in the academic community although it is of primary importance in most deployments. In particular, the sink has to be able to send packets to the nodes in the following scenarios:

- **Actuator:** Some motes may be able to have a physical action on the environment. They must consequently receive some commands, either from the sink or from another node [86];
- **Over-the-air reprogramming: Since the wireless industrial network** has to support multiple applications or to react after an event detection, some nodes may have to be reconfigured [87]. These commands are transmitted from the sink to the motes typically in multicast.

3) Localized: M2M traffic tends to be present in many of the current deployments such as in smart homes [88]. In this case, the considered topology does not rely on boarder-routers to collect the measurements and push them in a cloud. In fact, when data is not transmitted to a centralized entity, the response time and the energy consumption may be significantly reduced [89].

In Figure 11c, each actuator directly collects the measurements of a group of sensors and take autonomously a decision. On the contrary, in Figure 11b, the data flows do not traverse into the whole network. Such application is more scalable, it may multiplex more efficiently transmissions, without creating bottleneck nodes.

Unfortunately, the traffic models are very specific to each considered application. For instance, while a home automation switch may command a neighbouring light, a boiler may use the temperature measurements coming from the whole building.

B. Traffic Profile

Finally, in this Section, we also define the traffic profile for each traffic flow.

1) Constant Bit Rate (CBR): CBR flows represent the most common traffic model in which a data packet is generated periodically, possibly fragmented into several frames. This models well the smart metering scenario, where some meters measure a physical phenomenon and report regularly their measurements. A CBR flow is defined by its period, i.e., time interval between two packets and we often assume fixed length messages.

In Figure 12, we consider five nodes generating CBR flows. The inter-packet time may be fixed independently for each node. In our scenario, S1 and S3 have the same CBR period and generate their packets simultaneously. Thus, they will contend and possibly collide for each one of their transmissions. On the contrary, node S2 has the same period but the packet generation is asynchronous. Similarly, node S5 has a different CBR period and contends only for the first transmission. Consequently, we have to carefully consider fairness (see Section III-A2c) among the different flows.

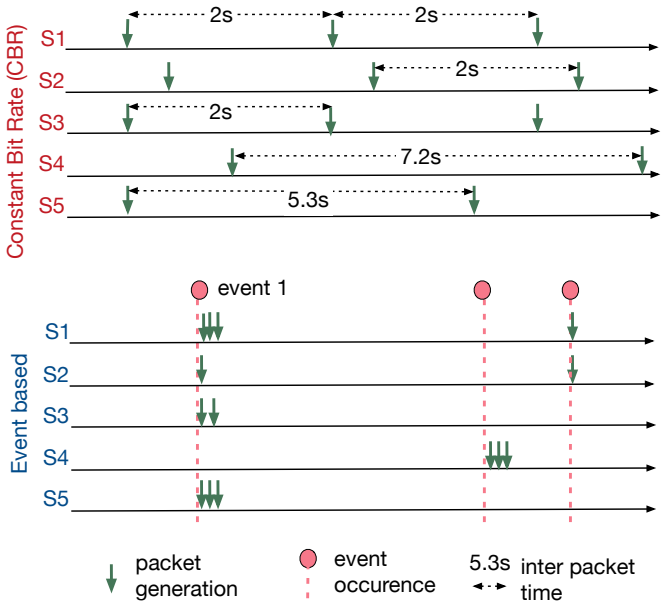


Fig. 12: Traffic profiles: Constant Bit Rate vs. Event based packet generation.

2) Event detection: the LLN infrastructure may also be employed for event detection [90]. Instead of executing computations in the cloud, low-power devices may implement more sophisticated techniques to process data. For instance, a node may transmit a measurement only if the temperature exceeds a specified threshold. By pushing the computations close to the producers, less transmissions are required and, thus, network lifetime is optimized.

The event-detection property has a direct impact on the traffic profile since packets are not uniformly generated. In Figure 12, the first event is detected by nodes S1, S2, S3 and S5 that all generate a burst of transmissions. We can remark that even if the global volume is similar to the CBR scenario, collisions may be more frequent since all nodes will intend to send their packets simultaneously.

Hereafter, we propose some potential approaches that can be adopted:

- **Real measurements** may be collected in a real-life deployment. These measurements can then be re-injected in simulations, *emulating* a real sensor or a real phenomenon.
- **Models** to mimic the physical environment are used in simulations. The simulator computes the list of simulated measurements acquired by each sensor. These measurements may trigger a packet generation. However, the models are very application specific, and require a very precise knowledge of the physical phenomenon, complicated to obtain in complex environments. Participatory sensing represents a promising way to collect data and to construct accurate models [91].

VI. Energy Consumption Models for Wireless Devices

The network lifetime is a major requirement because most of the participating nodes run on batteries. We already presented in Section III-A1a the different metrics that help to estimate energy consumption and network lifetime in general.

A. Residual Energy Estimation

Measuring accurately the residual energy is practically a complicated task [92]. Most techniques rely on measuring the battery voltage, (non linearly) correlated with the residual battery level. However, this correlation function depends on the battery model, the temperature, etc. [93] making an inaccurate estimation.

To estimate network lifetime would consequently require to execute the application until e.g. the first node depletion. For lifetime of months or even years, this approach is unrealistic. Furthermore, many algorithms would operate more efficiently if they are aware of the residual energy of each node, to e.g. change routing decisions for a better load balancing. Thus, most of the solutions rather try to estimate energy consumption.

B. Packet Based Estimation

Heinzelman *et al.* [94] propose to focus uniquely on the communication subsystem, because it represents the main source of energy consumption. The energy drained by a transmission takes into account:

- **The joule per bit** energy consumption;
- **Fixed energy** drained for any transmission, whatever the packet length is;
- **Distance:** The euclidean distance between the transmitter and the receiver. The authors consequently assume an ideal radio propagation, without noise and interference, with ideal radio chipsets.

Inversely, the energy to receive a packet is linear with the packet size. The model has to be parameterized according to the radio chipset.

Polastre *et al.* [95] also consider the energy for listening, sampling and receiving. The radio state may be represented by a state machine (sleep, TX, RX, etc.), and the transition duration has also to be considered. For instance, almost 2ms are required to initialize the radio before a reception when a CC1000 chipset was in sleeping mode.

Wang *et al.* [96] use such a model to derive the whole network lifetime, taking into account interference, multihop routes, channel acquisition, etc. However, such analytical models tend to under-estimate the impact of several environmental parameters (such as temperature, radio link quality, time variability, etc.)

Muller *et al.* [97] developed a specific circuit to measure the battery voltage, from a home-made mote. The authors combine this circuit with a packet-level method in order to estimate more accurately the energy consumption.

C. Transmission power adaptation

In topology control, a node selects the list of neighbours with which it will communicate. To reduce the level of interference and its energy consumption, it may adjust its transmission power so that the received signal strength is just

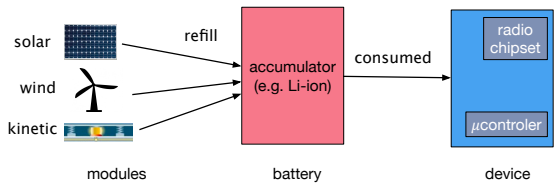


Fig. 13: Energy harvesting components.

above the threshold required to decode it correctly. Aziz *et al.* [98] present a comprehensive view of the different topology control techniques to prolong network lifetime.

D. Energy harvesting

Energy harvesting consists of incorporating a module into a device that collects energy from the environment [99]. A solar panel, a piezoelectric device or an anemometer provide energy to the

device, which stores often this energy in its battery (Figure 13). Energy harvesting requires to redesign the stack of protocols. Indeed, the routing decision should depend on the level of battery of the relay nodes [100].

A super capacitor helps to temporarily store the harvested energy [101]. Since it does not need a dedicated charging circuit, it presents a better conversion ratio. However, substantial leakage power losses need to be considered since they significantly impact efficiency.

The harvested energy depends on the source; solar energy obviously is ineffective during the night. The level of charge can be modelled by employing a discrete Markov Chain [102]. A transmission or reception *consumes* energy, while the harvesting source will fill the battery. Because the harvested power is often much smaller than the transmission/reception power, the communication is considered instantaneous.

VII. Available Tools for Performance Evaluation

In this Section, we detail a variety of tools and methods that can be utilized in order to evaluate the performance of different LLN applications, algorithms or protocols. Usually the researchers start with the numerical analysis, where they test their ideas in a more theoretical approach, based on mathematical formulas with realistic values. Next, the network simulators can be used so as to evaluate those theoretical assumptions, but with an essence of abstraction. However, there are some advanced simulators known as emulators, which are able to communicate with real network devices such to provide more realistic simulations of hardware aspects (e.g., processor performance during duty cycles, energy consumption). Furthermore, due to the current low cost of acquiring simple actuators, sensors, microprocessors and other modern IoT hardware, the research community has developed numerous open physical testbeds that allow for network experimentations. These testbeds can provide the ability to the scientists to expose their solutions to the most realistic conditions possible, prior to real deployment. Such deployments require to learn about available operating systems and supported platforms. This section thus provides an overview of existing emulators, which help to provide more accurate results by hosting specific lightweight OSs that run on top of specific target hardware. We discuss to what extent emulators may become a mandatory step for thorough evaluation of networking protocols.

A. Numerical Analysis

The simplest way to study the behaviour of an algorithm consists of replacing its different inputs by realistic values. For instance, a centralized algorithm can use a graph based model, to define which nodes (aka vertices) have a common radio (edge). To evaluate the dependency with the input parameters, we often use Monte Carlo simulations; different values are used and should lead to a similar behaviour. As an example, 20 source nodes should generate approximately as many collisions, wherever they are located.

Those numerical computations can be addressed with various programming languages, such as MATLAB, Python and GNU Octave, where each one of them have their pros and cons. The first difference someone can notice is that MATLAB requires to purchase a licence in order to utilize its core functions, while both Python and Octave are open and free programming languages. Furthermore, additional packages and libraries for MATLAB, such as SIMULINK, add extra charges in the overall cost.

Regarding their syntax, MATLAB and Octave are high-level languages. Also, Octave provides a MATLAB-like environment that facilitates the transition between the two languages. On the other hand, Python aims to be used as a full scale programming language which can support OS specific modules like process scheduling and multi-threading, as well as networking and databases. Consequently, programming in Python requires a longer learning curve to get familiar with its core libraries and programming flow, but it is easy to learn even from novice programmers. Additionally, a great advantage of Python is that it can support fast prototyping for different functionalities in LLN applications, considering some recent efforts from the community to develop smarter implementations of Python, like PyPy[6], as well as tiny microcontrollers that runs in Python[7]. A brief description of each language follows up.

1) MATLAB: MATLAB (MATrix LABoratory) is a software package and a high-level scripting language that enables high performance numerical computation and visualizations of new ideas. It represents one of the most popular software packages for scientific research. The mathematical framework can provide solutions on a broad field of mathematical problems such as matrix algebra, complex arithmetic, linear and non-linear systems, differential equations, signal processing etc.

2) GNU Octave: GNU Octave[8] is an interesting tool, initially published in 1993 and is compatible with Matlab

[6] http://pypy.org/
[7] https://micropython.org/
[8] https://www.gnu.org/software/octave/

scripts, . In particular, it is used to validate algorithms using the queueing theory and Markov chains[9].

3) Python: This language is now widely used for scientific computing, in particular with the SciPy library[10]. In order to evaluate network algorithms, the library NetworkX[11] provides routines for basic graph computation, simplifying the scientific development. For instance, scheduling algorithms (e.g. TASA [19]) often use Python to validate performance in an ideal graph.

4) Limitations: Numerical analysis represents the first step to validate an algorithm. However, results tightly depend on the accuracy of the models. Unfortunately, the most realistic models (interference, radio link quality) make the numerical analysis quickly intractable. Thus, another tool to validate performance should be employed in most of the scenarios.

B. Network Simulation

Large-scale testbeds are expensive to be developed and need great effort to be managed [103]. Therefore, network simulators have been designed with an aim to provide a software platform which can address the key aspects of the overall performance of different types of networks. However, it is almost impossible to duplicate the exact same conditions of real deployments. Simulations have been proved to not estimate correctly the impact of physical phenomenon [104].

On the other hand, simulators can be considered essential for exploring LLN applications, acting as a common ground for the scientists to test their ideas [105]. In order to effectively evaluate a study by employing simulation campaigns, it is important to have a good knowledge of the existing simulators and their capabilities.

Some examples of widely used simulators from the research community, include the ns2[23] ns3[24], OMNeT++[25], Riverbed Modeler[27], Qualnet[28] and WSNet[106]. However their individual capabilities and approaches on simulating the different details and types of networks, are quite distinct.

1) Proposed services:: the event-discrete simulators may propose some additional services to simplify the development.

- **Topology generation:** to mimic the different scenarios, we have to simulate different topologies (see section IV-A). The topology may be defined as a scenario with a GUI (e.g. riverbed), or via a configuration file (e.g. WSNet). To replay the same setup, flat configuration files should be preferred since they simplify the distribution, and can be easily converted into other formats;
- **Mobility models:** most simulators provide a way to control the trajectories for some of the nodes. Because of the discrete-event engine, the location of a device is updated periodically, simulating a movement.
- **Statistics collection:** to measure the performance, the devices have to be instrumented. Some of the simulators provide proprietary API to collect statistics (e.g. Riverbed, ns3). ns3 provides also a generic way to directly capture the different packets transmitted/received by the nodes. By analyzing this packet capture, we can compute e.g. the Packet Delivery Ratio, the end-to-end delay (see section III-A) using the same scripts as for experiments on real testbeds. Such method reduces the development costs when moving from simulation to experiments.
- **Visualization tools** are not required stricto sensu to measure the performance, but simplify greatly the debugging tasks. For instance, ns3 relies on PyViz [112] to debug a simulation, e.g. where a packet is dropped, the movement of a specific device. To our mind, offline visualization tools, which may exploit traces, should be preferred. They allow to replay the simulation, and to detect and localize a fault when an inconsistent result is obtained;
- **Clock drifts:** to the best of our knowledge, the simulators do not mimic clock drifts while they may have a significant impact on the performance [113]. Some additional modules or the engine have to be developed to quantify the impact of clock drifts.

2) Solutions: We detail here the most popular simulation tools used by the LLN community. Table II summarizes their main characteristics.

- **ns2** [23] (*"Network Simulator"*) is a discrete-event simulator mostly used for a research or educational purpose. Since it was primarily designed to simulate the Internet, it provides the full IP stack (TCP, IPv4, IPv6, etc.). Its *models* (protocols) are written in C, while Tcl/Tk scripts control the simulation (e.g. when a TCP flow starts).
  However, ns2 requires a long learning curve and advanced skills in order to conduct valuable and repeatable simulations. Besides, ns2 does not target specifically LLNs, and requires huge development effort to have a full LLN compliant stack.
- **ns3** [24] is the latest version of the Network Simulator. Instead of an evolution, the team has re-wrote from scratch the simulator, making it incompatible with its previous version. The ns-3 scenarios and models are implemented entirely in C++ with optional use of Python and PERL bindings.
  ns-3 provides an interesting hybrid mode, mixing emulation and simulation to focus on a specific physical phenomenon. Besides, it provides tools to measure the performance of the network by analyzing the packets captures by each device. This genericity is a strong asset: the same tools can be used for simulations and experiments, reducing the development costs, and making the analysis more reproducible.
- **OMNeT++** [25], is a component-based and discrete-event framework, which was primarily developed for building network simulators for wired and wireless networks. Support for specific domain networks, such as sensor networks has been provided through separate packages such as Castalia [114] WSN simulator. Furthermore, its core is totally implemented in C++, with tools for traffic visualisations, packet tracing and debugging.
  Most state of the art standards for LLNs are not na-

[9] http://www.moreno.marzolla.name/software/queueing/
[10] Scientific Computing Tools for Python, http://www.scipy.org
[11] High-productivity software for complex networks, https://networkx.github.io/

Simulator	Type[12]	Language	Models[13]							Specific features
			Mobility	IEEE802.15.4-2006	Bluetooth	DSME	TSCH	RPL	CoAP	
ns2	O	C & Tcl/Tk	y	y	y	n	n	n	n	parallel processing for scalable simulation
ns3	O	C	y	y	n	n	n	y	[107]	packet capture (libpcap) for statistics
omnet++	O	C & XML	y	y	n	n	n	[108]	n	
riverbed modeler	C	C	y	y	y	n	n	n	n	GUI for a smoother learning curve
Qualnet	C		y	y	n	n	n	[109]	n	Takes benefit from parallel architectures
WSNet	O	C & XML	y	[110]	n	n	n	[111]	n	dedicated for low-power nodes

TABLE II: Summarized characteristics of popular network simulators.

tively supported. For instance, an RPL implementation is provided in [108], but some of the features such as the Destination Advertisement Object (DAO) are not supported.

- **Riverbed Modeler** is a commercial, object-oriented, discrete-event network simulation software that is capable of operating in packet-level [27]. Riverbed Modeler provides a powerful GUI for the designing of simulation scenarios, which tends to reduce the learning curve for beginners. Recent versions also provide a large set of propagation models, with an implementation of the IEEE 802.15.4 and ZigBee standards.
  However, Riverbed Modeler does not target specifically the low-power devices. Thus, most of the protocols and standards for LLNs are not supported natively.
- **Qualnet** is a proprietary simulator, supporting parallel processing to make the simulation scalable [28]. It mimics real communications networks, enabling to model the whole propagation environment in a variety of complex situations (e.g. a collection of buildings for smart cities). It provides a library comprising a set of protocols (mainly Zigbee).
  As a descendant of GloMoSim[26], Qualnet simulator was rather designed for ad-hoc networks and its usage in LLN scenarios is not well recognized by the research community.
- **WSNet**
  WSNet [106] is an event-driven simulator dedicated to WSNs, which has been extensively evaluated [115]. It implements a large collection of radio propagation models, and the PHY layer can be modified easily (MIMO, modulation, multiple radio interfaces). A node may be mobile, and a physical environment may be simulated (e.g. fire detection).
  The simulator and its protocols are fully implemented in C and XML. Unfortunately, each protocol requires a specific implementation, and the current library and packages are quite limited.

3) Limits: The limits of simulations correspond to the limits of the simulation models. For instance, the most sophisticated radio propagation models should be preferred, but their complexity has an impact on the computation time. To maintain a reasonable computation time, we have consequently to consider only simple radio propagation models or a small number of nodes. Besides, Colesanti *et al.* [116] have highlighted that simulation results may differ significantly from real experimental results.

Most of the simulators are not specialized and target the large scale networks in general. Some of they provide a wireless library but with a very limited (and not up-to-date) set of protocols. Re-implementing the different standards and protocols increase the development costs, and the number of possible bugs. The reproducibility and the comparison of different implementations are also problematic.

The emulation/experimentation community (e.g., COOJA/Contiki, OpenSim/OpenWSN) is much more active. Most of the latest version of the IETF drafts and the latest standards are often available. Thus, we **recommend now to skip this simulation step to go directly to emulation, simulating only the PHY layer**. Possibly, simulation may still be used to study in depth a very specific phenomenon, and not the whole behavior of a full stack.

C. Testbeds

As it was previously exposed, testing and verifying new protocols and applications only over simulations may imply simplified assumptions, given the absence of accurate energy and radio simulation models, as well as considering the great complexity of real deployments.

Therefore, performing experiments with real hardware allows scientists to analyse performance of their applications under realistic environments. Indeed, due to the recent technological advances, the cost of such hardware has been reduced, thus, allowing the deployment of various large-scale controllable and manageable experimental testbeds.

Many testbeds are open to the research community for conducting their experiments with different characteristics, as they are summarized in Table III.

In order to ease the choice of the testbed and facilitate the experiments, we propose some guidelines for users to use repeatable setups and obtain reproducible results.

1) Characteristics: Existing testbeds present their own characteristics that should be carefully studied before opting for a given facility.

- **Hardware requirements:** They play a critical role for conducting realistic performance evaluations. The testbed of choice should correspond to the appropriate hardware requirements of the tested application in order to enable the researchers to investigate in depth its functionality prior to real deployment. The hardware parameters of a physical LLN platform include the network heterogeneity and scalability, where the underlying devices play different roles and reserve various resources. According to Yarvis et al. [117], heterogeneity can be distinguished into three types. The first type is the computational heterogeneity where some of the nodes have increased computational abilities, such as the sink nodes. Another type that could be considered is the link heterogeneity where some of the nodes may have wired interfaces in order to establish reliable communication links. The last type is the energy heterogeneity where the nodes may utilize different energy resources.
- **Radio environment and physical topology:** Radio propagation necessarily depends on the environment of a testbed (e.g., materials, thickness of ceiling and walls, density of plants in case of outdoor platform). Users may select a testbed whose deployment area presents the more similarities with the one of the target application. In addition, some facilities either offer only indoor nodes (e.g., Indriya, FIT IoT-lab) while others include outdoor devices (e.g., ORBIT, FLOCKLAB). Testbeds also greatly differ in terms of number of devices that are made available. As FLOCKLAB presents 40 sensors, TWIST or Indriya involve more than 100 motes, while FIT IoT-lab announces more than 2,500 nodes that can be booked for experiments. Once again, those characteristics should be studied carefully before choosing a platform. Note that some testbeds may allow end users to book only a subset of the available devices. Consequently, on some facilities, various experiments may co-exist and experimented solutions would then face noisy environments. This may be considered as a perfect environment for users aiming to test their solutions under unanticipated interferences.
- **Mobile nodes:** Many LLN applications involve mobile nodes that require special communicating schemes in order to interchange sensory data and information. Testing and executing mobile scenarios during an experimentation procedure requires to involve and combine advanced and intelligent technologies such as robots. Consequently, few of the widely popular open platforms do support mobility [118], even though more and more modern experimental facilities employ robotic and automation systems (e.g., Emulab, FIT IoT-LAB, KANSEI). Numerous challenges need to be addressed when having mobile robots in a testbed, namely, charging, remote administration and maintenance of the robots. Indeed, robots must be able to reach their docking stations automatically. They must also be able to follow the assigned trajectories. Thus, accurate positioning and path planning mechanisms with obstacle avoidance should be supported by such testbeds. Conversely, remote users must be able to interact with robots over reliable links (e.g., WiFi).

2) Proposed services: In addition to these characteristics, open testbeds may propose some services that would ease the setup of experiments while guaranteeing uninterrupted operations.

- **Maintenance and configuration:** Regular maintenance is appropriate to verify that the hardware and the software architecture of the testbed are still operational so as to manage effectively the experimental queue. Additionally, scheduled maintenance must take place in order to update the provided services and hardware components like the motes' batteries, such to optimize the functionality and extend the lifetime of the facility. End users would expect such information to be available in order to select the most appropriate facility. Furthermore, regarding the configuration of experiments, existing platforms present various tools that allow to design, conduct and analyse the experiments in a convenient and reliable way. The prime objective for end users is to finely configure their experimental setup. This includes the communication interfaces and protocols between the user and the underlying hardware in the different network layers, the scenario repeatability and possible simulation interoperability.
- **Radio link characterization:** In addition to the physical topology (i.e., location of nodes), testbeds may expose the radio topologies that can be tested. For instance, most of routing and MAC protocols require bidirectional links so that two nodes can exchange information (e.g., data and acknowledgement frames). Due to the realistic environments in which testbeds are deployed, such assumption may not hold. Thus, end users should be able to identify the different radio links and their associated properties (e.g., average bit error rate, distribution of radio link quality over time) before running their solution. Some dedicated tools would assist them to characterize the used topologies and their associated trust level. The latter would allow users to select the most appropriate nodes and links for testing their own solutions and to compare the obtained results against the existing ones.
- **Topology control:** Once the radio topology and link characterization are known to the users, the topology may be controlled. Some facilities would allow for topology control with either basic selection of used nodes or enhanced transmission power adaptations. Users may also be able to impact wireless transmission by inducing radio noise and interferences within the network. Some testbeds have incorporated such features (e.g., FIT IoT-lab). Even though a fine tuning of the jamming devices is required to reproduce observed realistic radio conditions, such a service could counterbalance the too idealistic environments in which some testbeds would be deployed. Additionally, the majority of LLN testbeds are equipped with commercially successful motes that provide different out-of-the-box functionalities such as to enable users to conveniently test various types of applications. Another option for addressing scalability and heterogeneity issues, is the federated model, which enables local experimental

Testbed	Nodes	Description
Emulab [119]	580 PC nodes with USRP 6 MICA2 robots 30 stationary MICA2	A combination of hardware and software tools. There are many instances of the Emulab framework deployed in more than two dozens sites around the world.
FIT IoT-LAB [**?**]	1144 MSP430-based motes 956 ARM Cortex M3 nodes 561 ARM Cortex-A8 nodes 108 open host nodes 50 wireless mesh routers six robots (127 planned)	Over 2700 indoor wireless sensor nodes spread across six different sites in France. Fixed and mobile nodes, embedding a variety of wireless sensors are available, with different processor architectures (MSP430, STM32 and Cortex-A8), and different wireless chips (802.15.4 PHY @ 800 MHz or 2.4 GHz). In addition, open nodes can receive custom wireless sensors for inclusion in IoT-LAB testbed.
FLOCKLAB [121]	30 Observers equipped with any four of Tmote Sky, OpenMote, MSP430-CCRF, TinyNode, Opal, and Iris motes	A mixed indoor/outdoor topology, able to support different services such as measuring power consumption and time accurate tracing and actuation.
Indriya [122]	139 TelosB	Based on MoteLab. The nodes are powered over the USB backchannel and equipped with light, temperature, acoustic, magnetometer, 2- axis accelerometer and infra-red sensors.
Intel Mirage [123]	97 MICA2 51 MICA2DOT motes	Based on a resource allocation system where the testbed resources are allocated, according to a repeated combinatorial auction. The motes are equipped with pressure, temperature, light and humidity sensors and powered over Ethernet.
KANSEI [124]	210 stationary nodes with Stargates, TmoteSky and Extreme Scale Motes 50 portable Trio motes five robots	Heterogeneous, hybrid experimental LLN laboratory that combines hardware motes, simulation engines and data generation devices
MAP [125]	32 static mesh routers 5 laptops and 16 PDAs	An experimental WMN laboratory. The testbed does not provide power consumption awareness.
MoteLab [126]	Fixed array of 30 MICAz 190 TelosB	One of the first open LLN testbeds, MySQL back-end server, a PHP web server, Java-based data logger and a Job Daemon for assigning tasks to the motes, Wall-powered with in-situ power measurement device in addition to temperature, humidity and light sensors.
NetEye [127]	130 TelosB motes 15 laptops	An open LLN experimental testbed equipped with light sensors and a mixed USB and Ethernet backchannel.
ORBIT [128]	400 nodes with more than 1,500 radio devices	A radio grid network testbed that consist of a remotely accessible indoor testbed, in addition to an outdoor trial network with mobile nodes.
Tutornet [129]	13 Stargates 91 TmoteSky 13 MICAz motes	A simple three-tiered, clustered LLN testbed
TWIST [130]	102 TmoteSky 102 eyesIFX	A central PostgreSQL server which is hierarchically organized in three layers, the servers, the super nodes and the sensor nodes. USB powered with light and temperature sensors. The super nodes are Network Link Storage Units.
UMass DieselNet [131]	40 buses with GPS devices HaCom Open Brick computer with 3 radios	A vehicular DTN of 40 public transport busses and various throwboxes that work as relays that promote the messages to the central repository.
WARPLab [132]	Up to16 WARP nodes	An experimental framework for experimentation of physical layer protocols controlled by a single PC by interfacing WARP nodes directly with MATLAB.

TABLE III: Summarized characteristics of popular LLN testbeds.

platforms to interconnect under a common framework in order to share their resources and provide more powerful evaluations, similar to WISEBED [133] and FIT-IoT [**?**] testbeds.

D. Which OS to choose?

Billions of sensor nodes, actuators smart meters and home appliances will be interconnected by wireless networks or Power Line Communication (PLC) [134], thus, giving birth to the IoT. Therefore, it is essential to have a lightweight operating systems that fulfills the requirements of such ultra low-power and extremely constrained, in terms of computing power and memory, devices.

In this subsection, we briefly introduce the operating systems for embedded and constrained devices that represent the most promising approaches towards realizing industrial (and not only) Low Power Lossy Networks. We focus on smart things with sufficient autonomous capabilities (i.e. not RFID tags) but which must also be energy aware (i.e. save energy). Note that most OSs (if not all) are written in the C programming language. Hereafter, we will detail the most popular operating systems. Hahm *et al.* [135] provide a very detailed and comprehensive survey about LLN operating systems.

1) TinyOS: TinyOS, based on event driven design, is so far one of the most used open source OSs for LLN scenarios [136]. It is written in the `nesC` programming language, a dialect of the C language, as a set of cooperating tasks and processes. TinyOS is designed for very low-power and constrained devices, i.e., 8 bit and 16 bit platforms, and is popular for its sophisticated design. Finally, it comes with BLIP network stack, an implementation of the 6LoWPAN stack.

2) Contiki OS: Contiki OS is one of the leading operating systems for embedded systems both in industry and in academia. Contiki is developed in C language, while some parts make use of macro-based abstractions (e.g., Protothreads [137]), and has been ported to a number of microcontroller architectures, including the Texas Instruments MSP430 and the Atmel AVR [142]. Contiki is running on the ESB platform [139] that uses MSP430 microcontroller with 2 kilobytes of RAM and 60 kilobytes of ROM running at 1 MHz. Furthermore, Contiki comes with a number of network stacks such as the uIP stack (with support for IEEE802.15.4-TSCH,

Emulator	Language	Models[14]								Specific features
		Mobility	IEEE802.15.4-2006	Bluetooth	DSME	TSCH	preamble sampling	RPL	CoAP	
Cooja	C	y	y	n	n	y	y	y	y	GUI interface
TOSSIM	C & nesc	n	y	n	n	n	y	y	y	complex learning curve for nesc
MSPSim	C	n	n	n	n	n	n	n	n	independent from the OS, reused by many platform and OS specific emulators
WSim/WSNet	C	y	y	n	n	n	n	y	n	independent from the OS, to be plugged with WSNet [106].
OpenWSN/ OpenSim	C & Python	n	n	n	n	y	n	y	y	GUI (web-based) interface

TABLE IV: Summarized characteristics of popular network emulators (all of them are opensource).

IPv6, 6LoWPAN, RPL, and CoAP) and the Rime stack, which provides a set of distributed programming abstractions.

3) FreeRTOS: FreeRTOS is developed since 2002 and is so far one of the most used open-source and Real-Time Operating Systems (RTOSs) for embedded and constrained devices. It has been ported to a large number of micro-controllers, and is the de-facto standard solution for micro-controllers and small micro-processors. FreeRTOS is written mostly in the C language, but there are a few assembly functions (i.e., mostly in architecture-specific scheduler routines). Even though, it does not come with its own full network stack, however, third-party network stacks can be employed for Internet connectivity.

4) RIOT: RIOT is designed and developed since 2012, by a growing, world-wide open-source community. RIOT is a microkernel-based RTOS with multi-threading support, while employing an architecture inherited from FireKernel [140]. The OS is written in C, while applications and libraries can also be implemented in C++. Furthermore, RIOT comes with several network stacks, such as the implementation of the full 6LoWPAN stack (i.e., the gnrc stack), a port of the 6TiSCH stack OpenWSN, and a port of the information centric networking stack CCN-lite[15].

5) OpenWSN: The OpenWSN project[16] was founded in 2010 (from the Berkeley Sensor & Actuator Center at UC Berkeley) and is developed since then by a growing open-source community that contributes to the Industrial Internet of Things (IIoT). OpenWSN comes with an implementation of a fully standards-based protocol stack (i.e., 6TiSCH network [141]) designed for capillary networks, rooted in the new IEEE802.15.4-2015 TSCH standard, and providing IPv6 connectivity to ultra-reliable and low-power industrial networks. Moreover, OpenWSN comprises a Board Support Package (BSP) i.e., a simple hardware abstraction.

In Table V, We here focus on most commonly available platforms, as summarized in Table III.

E. Emulation

To complete our presentation of the available tools with emulators since many solutions are tightly related with the OS and protocol stack to use. Only WSim and MSPsim are independent from the OS, but some key features (such as physical communications, mobility support, high-level description, etc.) are not emulated.

However, emulation is a very convenient tool for performance evaluation. The whole hardware platform is *emulated*, so that the same implementation may be used in testbeds or even in real-life. By debugging the emulated mote, we can also identify more efficiently the potential bugs.

Table IV illustrates a summary of the different characteristics and protocols supported by the major emulators for LLNs.

1) COOJA: is an open source and flexible simulator for the Contiki OS specialized for sensor nodes [142]. COOJA supports a large variety of hardware, including the most popular platforms (TelosB, Zolertia Z1, etc.) This emulator is written in Java and is supported consequently on mist of the OS.

2) MSPSim: MSPSim is an open source instruction set emulator able to simulate complete motes such as Tmote Sky, as well as custom motes based on Texas Instruments MSP430 microcontroller [143]. Extendibility is supported through a variety of available build-in implementations of different peripheral devices as components, which are further simulated based on a discrete-event approach.

3) TOSSIM: is a discrete event simulator and part of the TinyOS project, which is an embedded operating OS specialized for sensor networks [144]. Thus, TinyOS applications can be compiled directly into the TOSSIM framework. Furthermore, TOSSIM replaces the low-level components of a TinyOS system, such as the Analog-to-Digital Converter (ADC), the Master Clock, the EEPROM and several of the components in the radio stack in order to emulate their real behavior. It also provides a GUI that enables handy visualization, designing and debugging of running simulation scenarios in a controlled and repeatable environment.

4) WSim: is a platform simulator that was developed as part of the Worldsense framework [145]. It relies on cycle-based simulations using microprocessor instruction-driven timings. The simulator is able to perform a full simulation of hardware events that occur in the platform and to give back to the developer a precise timing analysis of the simulated software. FreeRTOS, Contiki and TinyOS operating systems have been

[15] http://ccn-lite.net
[16] http://openwsn.org/

Operating system	Latest release	Supported architectures				Stacks				RAM / ROM (min. required)	Real-Time support	Emulator
		Atmel AVR	PIC32	TI MSP430	STM32	Standard TCP/IP	6LoWPAN / RPL / CoAP		6TiSCH			
Contiki-NG	4.0 (Nov. 6th, 2017)	x	x	x	x	x	x			2kB / 30kB	partial	COOJA
FreeRTOS	10.0.0 (Nov. 28th, 2017)	x	x	x	x	x	x			1kB / 5kB	full	Windows FreeRTOS port
OpenWSN	1.8.0 (Oct. 5th, 2014)			x	x	x	x	x		3,7kB / 31kB	partial	OpenSim
RIOT	2017.10 (Oct. 27th, 2017)	x	x	x		x	x			1,5kB / 5kB	full	None official, even though RIOT OS executables can be simulated with Cooja.
TinyOS	2.1.2 (Aug. 20th, 2012)			x		x	x			1kB / 4kB	none	TOSSIM

TABLE V: Summarized characteristics of popular and open-source operating systems.

successfully tested on Worldsense. WSim can be used in standalone mode for debugging purpose, or interfaced with the WSNet simulator to perform the simulation of a complete sensor network.

5) OpenSim: is an open source emulator and simulator which was tailored to work with OpenWSN [33]. It is written in Python, and can be handled via a web-based interface to control the devices, push some commands, change the link quality, etc. Moreover, it can even mix emulated devices with real devices. It supports currently a very large variety of hardware platforms.

6) Conclusions: Emulating a device is CPU intensive, while emulating a large-scale network requires a powerful machine. Thus, emulation/simulation is limited to small scale topologies for prototyping to focus on a given phenomenon while large-scale evaluations may use a public testbed.

Emulators are often tied with a particular OS. Thus, the communication stack will probably guide the choice for the performance evaluation. For instance, COOJA seems the most relevant platform for preamble sampling protocols while OpenWSN is strongly recommended for a 6TiSCH stack since it integrates the last updates from the IETF Working Group.

VIII. Guidelines and Directions

A. Reproducibility

An experiment has to be *reproducible*, leading to the same results, when facing to the exactly same conditions [146]. With simulation, it should be sufficient to provide a detailed description of the simulation setup and the implementation of the algorithm/protocol. Unfortunately, such detailed description is very infrequent in the literature.

According to our perspective, performance evaluation should describe the following points:

- **Exhaustive description of the setup:** the authors have to describe without any ambiguity their performance evaluation setup (traffic, topology, models). The source code should be ideally distributed, or at least the binary (firmware) of the protocol's stack used for the performance evaluation;
- **Scenario Replay:** a set of scripts that permit to replay all the experiments should be provided, so that anyone can reply the same scenario on the same or different testbeds. For scenarios considering the mobility, the testbeds should provide a way to replay the same trajectory, with the same signal quality. This represents currently a key challenge since GPS is inaccurate for the indoor environments. Currently, even with perfect localization of all robots, trajectories are very difficult to replay, especially due to the odometer drifts.
- **A raw dataset** should be available in order to provide the whole set of results (before filtering). These measures have to be stored durably, to guarantee a free access in the next decades. This dataset can be easily re-interpreted later, using the latest, up-to-date scientific knowledge, like Raman *et al.* did with the MIT roofnet dataset [147].
- **A public repository** should collect all the traces, the implementation and the scripts. Crawdad [148] aims to collect a list of traces for different wireless networks (mainly mobility traces). This good practice would simplify the quantitative comparison against related work and would also allow other researchers to analyse the results in order to identify new phenomena/problems.

Regarding experimentations, we shall be able to replay the same experimental setup over stable and finely controlled hardware components. The real-world environment should be controllable to provide exactly the same conditions. This imposes to guarantee stability of hardware and environment components over time. Similarly to Guix-HPC[17], an effort to optimize GNU Guix for reproducible scientific workflows in High-Performance Computing (HPC), a future direction would consist in designing building blocks that could be configured and used within IoT testbeds (e.g., sets of topologies, mobility and traffic patterns, interference maps) in order to conduct repeatable setups and publish reproducible results.

B. Diversity and robustness

Performance evaluation may be very dependent on the conditions. The testbed or the simulation models have to be the most realistic as possible to predict the *actual* performance. An experimental validation is even not the panacea since different testbeds do not provide the same environmental conditions (e.g. indoor vs outdoor, etc).

[17] https://guix-hpc.bordeaux.inria.fr/

In order to make the performance evaluation robust, we have consequently to evaluate an algorithm that in the most diverse situations it will have to take into account the following aspects:

- **Hardware:** If a proposition is very closely related to the hardware used, we should proceed to the performance evaluation on top of different hardware (e.g. different radio chipsets to verify the presence of asymmetrical links);
- **Topology:** To verify that a solution performs well in *any* situation, we should study its behaviour in various conditions and to measure the impact of different parameters (e.g. density, scalability, traffic intensity);
- **External Interference:** To make the experiments reproducible, some nodes (with the same or different technologies) may be used to generate an interfering signal. Such experimental setup would maximize the reproducibility, while estimating the robustness to external interference [7].

Federated testbeds should make such comparison easier, letting researchers to reserve several different testbeds [149].

C. Long term evaluation

While low power lossy networks expect to be deployed for a long-term use, most of the performance evaluations focus surprisingly on experiments which last for a few minutes. Many papers even do not specify the duration of their experiments / simulations [12].

To demonstrate the relevance of LLNs, solutions have been deployed and monitored. Zebranet [150] and Sensorscope [151] provide results over one month. However, they often consist in real deployments, where the application is the main concern.

To more accurately capture the actual performance of a low power lossy network, we need to study more in depth the following points:

- **Convergence**: Many papers propose to discard the first minutes of the experiments / simulations and focus on the steady state only. However, the bootstrapping phase plays a major role in the convergence time and thus appears of primary importance as well. For instance, with RPL it is common that many routes may change initially, consuming a large overhead [152]. Thus, we have to study more carefully this phase in order to accelerate the convergence of the solutions: waiting for a few hours may be unacceptable for many deployments.
- **Long-term** evaluations: we need to start experiments which should last at least for few days. Rare phenomena may have significant impact to the performance of the network, but can only be captured if the experiment lasts long enough to capture all the situations. Furthermore, the characteristics of the network may even be time-variant (see Section IV-A3c).

D. Re-injection of real measures

Simulations are often far from the reality, since they are based on simple models, concerning for instance the PHY layer. Unfortunately, the interference model deeply impacts the performance [74]. Even worse, the behavior of the network has been proved to deeply depend on the simulator for wireless networks [153]. On the contrary, experiments are complicated to reproduce and interpret, because may external factors (e.g., external interference) impact the performance of the network. Emulation seems a promising way to simplify the performance evaluation, but it relies also on a synthetic PHY model to mimic the transmissions.

A promising approach seems to combine experiments through a testbed with simulation or emulation: the experiments provide the PHY measures concerning the success of each transmission, information re-injected in simulation or emulation to mimic the behavior of a real radio medium. The same measures may be re-injected to make the evaluation reproducible.

A few initiatives already proposed a free access to such datasets:

- the packet delivery ratio for 55 devices, over 16 frequencies at USC[18] [154].
- aggregated statistics (e.g., Packet Delivery Ratio, mean LQI) for a testbed with 10 nodes executing RPL during one day[19].
- packet delivery statistics (i.e. packet received or not) for 267 radio links, over the 16 IEEE802.15.4 channels, during 1.5 hour[20].

Having the raw (non aggregated) results should be preferred to be the most generic as possible.

Unfortunately, probing each radio link individually to estimate the level of interference, and the probability of collisions is very expensive. Indeed, we have to test each subset of radio links iteratively, with a train of probe packets [7]. Such an exhaustive method is quickly inapplicable, and some heuristics have to be proposed to mimic actually a real deployment.

E. Definition of Scenarii and Benchmarking

For low power and lossy networks, we have a profusion of solutions which are evaluated in very different conditions [12]. Unfortunately, multiplying the conditions complicates significantly the interpretation and the comparison. The IETF ROLL WG has defined in the past the requirements for smart building [4], smart homes [155] and industrial [156] applications. Similarly, the ETSI has defined the traffic profile for many applications in smart cities, detailing the requirements in delay, bandwidth, etc. [157].

We need to define clearly some scenarii:

- we need to define scenarii that each researcher should use for a collection of different applications. The scenario should be entirely defined (topology, link quality, variability, etc.) so that the different results are reproducible and so that we can compare easily different algorithms / protocols;
- we need to collect real data on large-scale testbeds corresponding to these scenarii. These measures can then

[18] http://anrg.usc.edu/www/tutornet/
[19] https://github.com/apanouso/wsn-indfeat-dataset
[20] https://github.com/ftheoleyre/fitiotlab-multichannel-dataset

be re-injected to replay the same experiment, so that emulation can then be used to automatize the comparisons (cf section VIII-D).

IX. Conclusions

Validation and verification during the development of protocols are a matter of prime importance. A large variety of practical and theoretical tools has been developed to assist researchers to assess the performance of their solutions. This tutorial aims at providing guidelines and a list of good practices for researchers involved in evaluating the performance of protocols for low-power and lossy networks. Several papers already provide a comprehensive view of existing testbeds and network simulators, as well as describing some of their individual pitfalls.

Thus, we here adopted an approach focused rather on the methodology and in particular what should a researcher take into consideration when evaluating performance of an algorithm. Therefore, we reviewed various parameters that should be considered during such performance evaluation.

Furthermore, we detailed the general approach adopted to evaluate the performance of networking protocols (i.e., layers 2 and 3) for low-power and lossy networks. In particular, we have tried to provide detailed information that could help answering questions like "How should a performance evaluation be started?" or "What are the different choices provided to evaluate performance and the different steps to be followed in order to definitely validate a contribution?". Therefore, we especially surveyed numerous models and tools that are available to the research community.

Typically, such a tutorial article makes sense for any researcher (i.e., MSc or PhD student, engineer) starting in this research area. By reading this paper, one could avoid the common pitfalls that most of the researchers face when they attempt to carry out a performance evaluation. The recent huge attention drawn on the reproducibility of the related research domain is nothing else but the willingness to adopt a methodical approach by considering all the different key aspects of LLNs.

References

[1] J. M. Schleicher, M. Vögler, S. Dustdar, and C. Inzinger, "Application architecture for the internet of cities: Blueprints for future smart city applications," *IEEE Internet Computing*, vol. 20, pp. 68–75, Nov 2016.

[2] B. Ahlgren, M. Hidell, and E. C. H. Ngai, "Internet of things for smart cities: Interoperability and open data," *IEEE Internet Computing*, vol. 20, pp. 52–56, Nov 2016.

[3] B. Zhou, J. Cao, X. Zeng, and H. Wu, "Adaptive traffic light control in wireless sensor network-based intelligent transportation system," in *2010 IEEE 72nd Vehicular Technology Conference - Fall*, pp. 1–5, Sept 2010.

[4] J. Martocci, P. D. Mil, N. Riou, and W. Vermeylen, "Building automation routing requirements in low-power and lossy networks," RFC 5687, IETF, 2010.

[5] G. Z. Papadopoulos, J. Beaudaux, A. Gallais, T. Noel, and G. Schreiner, "Adding value to WSN simulation using the IoT-LAB experimental platform," in *International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, IEEE, 2013.

[6] G. Z. Papadopoulos, A. Gallais, G. Schreiner, and T. Noël, "Importance of Repeatable Setups for Reproducible Experimental Results in IoT," in *Proceedings of the 13th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN '16)*, pp. 51–59, 2016.

[7] V. Kotsiou, G. Z. Papadopoulos, P. Chatzimisios, and F. Theoleyre, "Is Local Blacklisting Relevant in Slow Channel Hopping Low-Power Wireless Networks?," in *Proceedings of the IEEE International Conference on Communications (ICC)*, 2017.

[8] G. Gaillard, D. Barthel, F. Theoleyre, and F. Valois, "Service level agreements for wireless sensor networks: A wsn operator's point of view," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, pp. 1–8, May 2014.

[9] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler, "Standardized protocol stack for the internet of (important) things," *IEEE Communications Surveys Tutorials*, vol. 15, pp. 1389–1406, Third 2013.

[10] V. Karagiannis, P. Chatzimisios, F. Vázquez-Gallego, and J. Alonso-Zarate, "A survey on application layer protocols for the internet of things," *Transaction on IoT and Cloud Computing,*, vol. 1, January 2015.

[11] R. C. Martin, *Agile Software Development: Principles, Patterns, and Practices*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2003.

[12] G. Z. Papadopoulos, K. Kritsis, A. Gallais, P. Chatzimisios, and T. Noel, "Performance evaluation methods in ad hoc and wireless sensor networks: A literature study," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 122–128, 2016.

[13] I. Stojmenovic, "Simulations in Wireless Sensor and Ad Hoc Networks: Matching and Advancing Models, Metrics, and Solutions," *IEEE Communications Magazine*, vol. 46, no. 12, pp. 102–107, 2008.

[14] K. Langendoen, A. Baggio, and O. Visser, "Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture.," *IPDPS 2006*, p. 8 pp., 2006.

[15] S. Lohs, J. Nolte, G. Siegemund, and V. Turau, "Self-stabilization - a mechanism to make networked embedded systems more reliable?," in *Symposium on Reliable Distributed Systems (SRDS)*, pp. 317–326, IEEE, Sept 2016.

[16] V. Vazirani, *Approximation Algorithms*. Springer, 2002.

[17] P. Bremaud, *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer, 2008.

[18] R. Soua, P. Minet, and E. Livolant, "Modesa: An optimized multichannel slot assignment for raw data convergecast in wireless sensor networks," in *Performance Computing and Communications Conference (IPCCC), 2012 IEEE 31st International*, pp. 91–100, Dec 2012.

[19] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and G. Boggia, "Traffic aware scheduling algorithm for reliable low-power multi-hop ieee 802.15.4e networks," in *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*, pp. 327–332, Sept 2012.

[20] A. Keshavarzian, H. Lee, and L. Venkatraman, "Wakeup scheduling in wireless sensor networks," in *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '06, (New York, NY, USA), pp. 322–333, ACM, 2006.

[21] X. Yin and X. Cheng, *Propagation Channel Characterization, Parameter Estimation, and Modeling for Wireless Communications*. Wiley & Sons, 2016.

[22] D. Djenouri and M. Bagaa, "Synchronization protocols and implementation issues in wireless sensor networks: A review," *IEEE Systems Journal*, vol. 10, pp. 617–627, June 2016.

[23] "Network simulator 2 (ns-2)." https://www.isi.edu/nsnam/ns/.

[24] "Network simulator 3 (ns-3)." http://www.isi.edu/nsnam/ns/, 2011.

[25] "Omnet++." https://omnetpp.org/.

[26] X. Zeng, R. Bagrodia, and M. Gerla, "Glomosim: A library for parallel simulation of large-scale wireless networks," *SIGSIM Simul. Dig.*, vol. 28, pp. 154–161, July 1998.

[27] "Riverbed modeler." https://www.riverbed.com.

[28] "The fastest, most scalable network modeling platform." http://web.scalable-networks.com/qualnet-network-simulator.

[29] L. Yan and N. McKeown, "Learning networking by reproducing research results," *SIGCOMM Comput. Commun. Rev.*, vol. 47, pp. 19–26, May 2017.

[30] C. Haas, J. Wilke, and V. Stöhr, "Realistic simulation of energy consumption in wireless sensor networks," in *European Conference Wireless Sensor Networks (EWSN)*, (Trento, Italy), pp. 82–97, 2012.

[31] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *Proceedings of the 4th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 364–369, 2005.

[32] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-Level Sensor Network Simulation with COOJA," in *Proceedings of the 31st Annual IEEE International Conference on Local Computer Networks (LCN)*, pp. 641–648, 2006.

[33] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister, "Openwsn: a standards-based low-power wireless development environment," *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 5, pp. 480–493, 2012.

[34] L. M. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, pp. 1548–1557 vol.3, 2001.

[35] I. Dietrich and F. Dressler, "On the lifetime of wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, pp. 1–39, Feb. 2009.

[36] D. Tian and N. D. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, WSNA '02, (New York, NY, USA), pp. 32–41, ACM, 2002.

[37] M. Saleem, I. Ullah, and M. Farooq, "Beesensor: An energy-efficient and scalable routing protocol for wireless sensor networks," *Information Sciences*, vol. 200, pp. 38 – 56, 2012.

[38] G. Z. Papadopoulos, V. Kotsiou, A. Gallais, P. Chatzimisios, and T. Noel, "Low-Power Neighbor Discovery for Mobility-Aware Wireless Sensor Networks," *Elsevier Ad Hoc Networks*, vol. 48, pp. 66–79, 2016.

[39] P. Huang, L. Xiao, S. Soltani, M. W. Mutka, and N. Xi, "The evolution of mac protocols in wireless sensor networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, pp. 101–120, First 2013.

[40] G. Z. Papadopoulos, J. Beaudaux, A. Gallais, and T. Noel, "T-AAD: Lightweight Traffic Auto-ADaptations for Low-power MAC Protocols," in *Proceedings of the 13th IEEE IFIP Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, pp. 79–86, 2014.

[41] A. Dunkels, "The contikimac radio duty cycling protocol," Tech. Rep. T2011:13, SICS, 2011.

[42] A. Mavromatis, G. Z. Papadopoulos, X. Fafoutis, A. Elsts, G. Oikonomou, and T. Tryfonas, "Impact of Guard Time Length on IEEE 802.15.4e TSCH Energy Consumption," in *Proceedings of the 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–3, 2016.

[43] G. Z. Papadopoulos, A. Mavromatis, X. Fafoutis, N. Montavont, R. Piechocki, T. Tryfonas, and G. Oikonomou, "Guard Time Optimisation and Adaptation for Energy Efficient Multi-hop TSCH Networks," in *Proceedings of the IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pp. 301–306, 2016.

[44] "IEEE Standard for Low-Rate Wireless Personal Area Networks (LR-WPANs)." IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011), April 2016.

[45] L. Gu and J. A. Stankovic, "Radio-triggered wake-up for wireless sensor networks," *Real-Time Systems*, vol. 29, no. 2, pp. 157–182, 2005.

[46] D. Giustiniano, D. Malone, D. J. Leith, and K. Papagiannaki, "Measuring transmission opportunities in 802.11 links," *IEEE/ACM Transactions on Networking*, vol. 18, pp. 1516–1529, Oct 2010.

[47] F. Hermans, O. Rensfelt, T. Voigt, E. Ngai, L. A. Norden, and P. Gunningberg, "Sonic: Classifying interference in 802.15.4 sensor networks," in *Information Processing in Sensor Networks (IPSN), 2013 ACM/IEEE International Conference on*, pp. 55–66, April 2013.

[48] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," Technical Report DEC-TR-301, Digital Equipment Corporation, 1984.

[49] F. Tobagi and L. Kleinrock, "Packet switching in radio channels: Part ii - the hidden terminal problem in carrier sense multiple-access and the busy-tone solution," *IEEE Transactions on Communications*, vol. 23, pp. 1417–1433, December 1975.

[50] R. P. Liu, Z. Rosberg, I. B. Collings, C. Wilson, A. Y. Dong, and S. Jha, "Overcoming radio link asymmetry in wireless sensor networks," in *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*, pp. 1–5, Sept 2008.

[51] G. Z. Papadopoulos, J. Beaudaux, A. Gallais, P. Chatzimisios, and T. Noel, "Toward a Packet Duplication Control for Opportunistic Routing in WSNs," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, pp. 94–99, 2014.

[52] S. Biswas and R. Morris, "Exor: Opportunistic multi-hop routing for wireless networks," in *SIGCOMM*, pp. 133–144, 2005.

[53] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados, "Energy-efficient routing protocols in wireless sensor networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, pp. 551–591, Second 2013.

[54] T. Winter, et al., "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," rfc 6550, IETF, 2012.

[55] T. Clausen, et al., "The lightweight on-demand ad hoc distance-vector routing protocol - next generation (loadng)," draft 15, IETF, 2016.

[56] H. Huang, H. Yin, Y. Luo, X. Zhang, G. Min, and Q. Fan, "Three-dimensional geographic routing in wireless mobile ad hoc and sensor networks," *IEEE Network*, vol. 30, pp. 82–90, March 2016.

[57] M. O. Farooq, C. J. Sreenan, K. N. Brown, and T. Kunz, "Design and analysis of rpl objective functions for multi-gateway ad-hoc low-power and lossy networks," *Ad Hoc Networks*, vol. 65, pp. 78 – 90, 2017.

[58] A. A. K. Somappa, K. Øvsthus, and L. M. Kristensen, "An industrial perspective on wireless sensor networks: A survey of requirements, protocols, and challenges," *IEEE Communications Surveys Tutorials*, vol. 16, pp. 1391–1412, Third 2014.

[59] P. Kampstra, "Beanplot: A boxplot alternative for visual comparison of distributions," *Journal of Statistical Software*, 2008.

[60] F. Theoleyre and G. Z. Papadopoulos, "Experimental Validation of a Distributed Self-Configured 6TiSCH with Traffic Isolation in Low Power Lossy Networks ," in *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, 2016.

[61] S. Chessa and P. Santi, "Crash faults identification in wireless sensor networks," *Computer Communications*, vol. 25, no. 14, pp. 1273 – 1282, 2002.

[62] M. Eskola and T. Heikkilä, "Classification of radio channel disturbances for industrial wireless sensor networks," *Ad Hoc Networks*, vol. 42, pp. 19 – 33, 2016.

[63] O. Iova, F. Theoleyre, and T. Noel, "Stability and efficiency of rpl under realistic conditions in wireless sensor networks," in *International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 2098–2102, IEEE, Sept 2013.

[64] M. Senel, K. Chintalapudi, D. Lal, A. Keshavarzian, and E. J. Coyle, "A kalman filter based link quality estimation scheme for wireless sensor networks," in *Global Telecommunications Conference (GLOBECOM)*, pp. 875–880, IEEE, Nov 2007.

[65] E. Schiller, P. Starzetz, F. Theoleyre, and A. Duda, "Properties of greedy geographical routing in spontaneous wireless mesh networks," in *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pp. 4941–4945, Nov 2007.

[66] F. Tobagi and L. Kleinrock, "Packet switching in radio channels: Part ii–the hidden terminal problem in carrier sense multiple-access and the busy-tone solution," *IEEE Transactions on Communications*, vol. 23, pp. 1417–1433, Dec 1975.

[67] C. Chaudet, D. Dhoutaut, and I. G. Lassous, "Performance issues with ieee 802.11 in ad hoc networking," *IEEE Communications Magazine*, vol. 43, pp. 110–116, July 2005.

[68] O. Iova, F. Theoleyre, and T. Noel, "Using multiparent routing in {RPL} to increase the stability and the lifetime of the network," *Ad Hoc Networks*, vol. 29, pp. 45 – 62, 2015.

[69] F. Stann, J. Heidemann, R. Shroff, and M. Z. Murtaza, "Rbp: Robust broadcast propagation in wireless networks," in *International Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 85–98, ACM, 2006.

[70] C. H. Lo and N. Ansari, "The progressive smart grid system from both power and communications aspects," *IEEE Communications Surveys Tutorials*, vol. 14, pp. 799–821, Third 2012.

[71] R. D. Gomes, D. V. Queiroz, A. C. L. Filho, I. E. Fonseca, and M. S. Alencar, "Real-time link quality estimation for industrial wireless sensor networks using dedicated nodes," *Ad Hoc Networks*, vol. 59, pp. 116 – 133, 2017.

[72] Y. Chen and A. Terzis, "On the implications of the log-normal path loss model: An efficient method to deploy and move sensor motes," in *Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 26–39, ACM, 2011.

[73] A. Anglès-Vázquez, X. Vilajosana-Guillèn, J. López-Vicario, A. Morell-Pérez, P. Tuset-Peiró, and I. Vilajosana-Guillèn, "Generic empiric propagation model for low power wireless networks operating at the 868 mhz band in smart cities," *IET Microwaves, Antennas Propagation*, vol. 8, no. 14, pp. 1143–1153, 2014.

[74] A. Iyer, C. Rosenberg, and A. Karnik, "What is the right model for wireless channel interference?," *IEEE Transactions on Wireless Communications*, vol. 8, pp. 2662–2671, May 2009.

[75] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis, "The β-factor: Measuring wireless link burstiness," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, (New York, NY, USA), pp. 29–42, ACM, 2008.

[76] A. Cerpa, J. L. Wong, M. Potkonjak, and D. Estrin, "Temporal Properties of Low Power Wireless Links: Modeling and Implications on Multi-Hop Routing," in *International Symposium on Mobile Ad*

Hoc Networking and Computing (MOBIHOC), (New York, New York, USA), pp. 414–425, ACM Press, 2005.

[77] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Models and solutions for radio irregularity in wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 2, pp. 221–262, May 2006.

[78] J. Padhye, S. Agarwal, V. N. Padmanabhan, L. Qiu, A. Rao, and B. Zill, "Estimation of link interference in static multi-hop wireless networks," in *SIGCOMM Conference on Internet Measurement (IMC)*, pp. 28–28, ACM, 2005.

[79] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," *Wireless Networks*, vol. 8, pp. 153–167, March 2002.

[80] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 11, pp. 6–28, Dec 2004.

[81] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Netw.*, vol. 11, pp. 2–16, Feb. 2003.

[82] P. Thubert, "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4." draft-ietf-6tisch-architecture-11, July 2017.

[83] C.-Y. Wan, S. B. Eisenman, A. T. Campbell, and J. Crowcroft, "Siphon: Overload traffic management using multi-radio virtual sinks in sensor networks," in *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, SenSys '05, (New York, NY, USA), pp. 116–129, ACM, 2005.

[84] G. Z. Papadopoulos, N. Pappas, A. Gallais, T. Noel, and V. Angelakis, "Distributed Adaptive Scheme for Reliable Data Collection in Fault Tolerant WSNs," in *Proceedings of the 2nd IEEE World Forum on Internet of Things (WF-IoT)*, pp. 116–121, 2015.

[85] A. Karnik, A. Iyer, and C. Rosenberg, "Throughput-optimal configuration of fixed wireless networks," *IEEE/ACM Trans. Netw.*, vol. 16, pp. 1161–1174, Oct. 2008.

[86] J. Chen, X. Cao, P. Cheng, Y. Xiao, and Y. Sun, "Distributed collaborative control for industrial automation with wireless sensor and actuator networks," *IEEE Transactions on Industrial Electronics*, vol. 57, pp. 4219–4230, Dec 2010.

[87] M. Szczodrak, O. Gnawali, and L. P. Carloni, "Dynamic reconfiguration of wireless sensor networks to support heterogeneous applications," in *2013 IEEE International Conference on Distributed Computing in Sensor Systems*, pp. 52–61, May 2013.

[88] M. Chen, J. Wan, S. Gonzalez, X. Liao, and V. C. M. Leung, "A survey of recent developments in home m2m networks," *IEEE Communications Surveys Tutorials*, vol. 16, pp. 98–114, First 2014.

[89] C. M. de Farias, L. Pirmez, F. C. Delicato, P. F. Pires, W. Li, A. Y. Zomaya, E. N. de L. F. Jorge, and R. Juarez-Ramirez, "Grown: A control and decision system for smart greenhouses using wireless sensor networks," in *Australasian Computer Science Week Multiconference (ACSW)*, pp. 48:1–48:8, ACM, 2017.

[90] D. C. Harrison, W. K. G. Seah, and R. Rayudu, "Rare event detection and propagation in wireless sensor networks," *ACM Comput. Surv.*, vol. 48, pp. 58:1–58:22, Mar. 2016.

[91] L. Wang, D. Zhang, Y. Wang, C. Chen, X. Han, and A. M'hamed, "Sparse mobile crowdsensing: challenges and opportunities," *IEEE Communications Magazine*, vol. 54, pp. 161–167, July 2016.

[92] V. Shnayder, M. Hempstead, B.-r. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, (New York, NY, USA), pp. 188–200, ACM, 2004.

[93] J. Kim and D. K. Noh, "Voltage-based estimation of residual battery energy in wireless sensor systems," in *SENSORS*, pp. 1–4, IEEE, Nov 2013.

[94] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, pp. 10 pp. vol.2–, Jan 2000.

[95] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, (New York, NY, USA), pp. 95–107, ACM, 2004.

[96] Q. Wang, M. Hempstead, and W. Yang, "A realistic power consumption model for wireless sensor network devices," in *Communications Society on Sensor and Ad Hoc Communications and Networks (SECON)*, vol. 1, pp. 286–295, IEEE, Sept 2006.

[97] I. Müller, J. Winter, C. Pereira, V. Brusamarello, and J. C. Netto, "Energy consumption estimation for tdma-based industrial wireless sensor networks," in *International Conference on Industrial Informatics (INDIN)*, pp. 625–630, IEEE, July 2016.

[98] A. A. Aziz, Y. A. Sekercioglu, P. Fitzpatrick, and M. Ivanovich, "A survey on distributed topology control techniques for extending the lifetime of battery powered wireless sensor networks," *IEEE Communications Surveys Tutorials*, vol. 15, pp. 121–144, First 2013.

[99] W. Ejaz, M. Naeem, A. Shahid, A. Anpalagan, and M. Jo, "Efficient energy management for the internet of things in smart cities," *IEEE Communications Magazine*, vol. 55, pp. 84–91, January 2017.

[100] Y.-H. Chen, B. Ng, W. K. Seah, and A.-C. Pang, "Modeling and analysis: Energy harvesting in the internet of things," in *International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pp. 156–165, ACM, 2016.

[101] A. S. Weddell, G. V. Merrett, T. J. Kazmierski, and B. M. Al-Hashimi, "Accurate supercapacitor modeling for energy harvesting wireless sensor nodes," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 58, pp. 911–915, Dec 2011.

[102] A. Biason and M. Zorzi, "On the effects of battery imperfections in an energy harvesting device," in *International Conference on Computing, Networking and Communications (ICNC)*, pp. 1–7, IEEE, Feb 2016.

[103] A. Stajkic, M. D. Abrignani, C. Buratti, A. Bettinelli, D. Vigo, and R. Verdone, "From a real deployment to a downscaled testbed: A methodological approach," *IEEE Internet of Things Journal*, vol. 3, pp. 647–657, Oct 2016.

[104] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott, "Experimental evaluation of wireless simulation assumptions," in *International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pp. 78–82, ACM, 2004.

[105] I. Stojmenovic, "Simulations in Wireless Sensor and Ad Hoc Networks: Matching and Advancing Models, Metrics, and Solutions," *IEEE Communications Magazine*, vol. 46, no. 12, pp. 102–107, 2008.

[106] "Wsnet." http://wsnet.gforge.inria.fr/.

[107] "ns3-coap." https://github.com/maesoser/ns3-coap.

[108] H. Kermajani and C. Gomez, "On the network convergence process in rpl over ieee 802.15.4 multihop networks: Improvement and trade-offs," *Sensors*, vol. 14, no. 7, pp. 11993–12022, 2014.

[109] V. Kathuria, G. Mohanasundaram, and S. R. Das, "A simulation study of routing protocols for smart meter networks," in *International Conference on Smart Grid Communications (SmartGridComm)*, pp. 384–389, IEEE, Oct 2013.

[110] N. Abdeddaïm and F. Theoleyre, "Implementation of a WSNet Module to Simulate the IEEE 802.15.4 Beacon-Enabled Mode in Multihop Topologies," tech. rep., HAL, 2011. https://hal.archives-ouvertes.fr/hal-00590853.

[111] L. B. Saad, C. Chauvenet, and B. Tourancheau, "Simulation of the rpl routing protocol for ipv6 sensor networks: two cases studies," in *SENSORCOMM*, 2011. https://hal.inria.fr/hal-00647869.

[112] "Pyviz - ns3." https://www.nsnam.org/wiki/PyViz.

[113] M. P. Uwase, M. Bezunartea, J. Tiberghien, J. M. Dricot, and K. Steenhaut, "Experimental comparison of radio duty cycling protocols for wireless sensor networks," *IEEE Sensors Journal*, vol. 17, pp. 6474–6482, Oct 2017.

[114] Castalia, "Castalia - Wireless Sensor Network Simulator." https://castalia.forge.nicta.com.au/index.php/en/index.html, 2013.

[115] E. Ben Hamida *et al.*, "On the Complexity of an Accurate and Precise Performance Evaluation of Wireless Networks using Simulations," in *MSWiM*, ACM, 2008.

[116] U. M. Colesanti, C. Crociani, and A. Vitaletti, "On the accuracy of omnet++ in the wireless sensor networks domain: simulation vs. testbed," in *Proceedings of the 4th ACM workshop on performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pp. 25–31, ACM, 2007.

[117] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh, "Exploiting heterogeneity in sensor networks," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 2, pp. 878–890 vol. 2, March 2005.

[118] A.-S. Tonneau, N. Mitton, and J. Vandaele, "How to choose an experimentation platform for wireless sensor networks? a survey on static and mobile wireless sensor network experimentation facilities," *Ad Hoc Netw.*, vol. 30, pp. 115–127, July 2015.

[119] University of Utah, "Emulab - Network Emulation Testbed." http://www.emulab.net/index.php3, 2017.

[120] O. Fambon, E. Fleury, G. Harter, R. Pissard-Gibollet, and F. Saint-Marcel, "Fit iot-lab tutorial: hands-on practice with a very large scale testbed tool for the internet of things," in *UbiMob*, 2014.

[121] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel, "Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems," in *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, IPSN '13, (New York, NY, USA), pp. 153–166, ACM, 2013.

[122] M. Doddavenkatappa, M. C. Chan, and A. L. Ananda, "Indriya: A low-cost, 3d wireless sensor network testbed," in *International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TRIDENTCOM)*, (Shanghai, China), pp. 302–316, EAI, 2012.

[123] B. N. Chun, P. Buonadonna, A. AuYoung, C. Ng, D. C. Parkes, J. Shneidman, A. C. Snoeren, and A. Vahdat, "Mirage: A microeconomic resource allocation system for sensornet testbeds," in *The Second IEEE Workshop on Embedded Networked Sensors, 2005. EmNetS-II.*, pp. 19–28, May 2005.

[124] A. Arora, E. Ertin, R. Ramnath, M. Nesterenko, and W. Leal, "Kansei: a high-fidelity sensing testbed," *IEEE Internet Computing*, vol. 10, pp. 35–47, March 2006.

[125] Purdue University, "Purdue University Wireless Mesh Network Testbed." https://engineering.purdue.edu/MESH, 2017.

[126] G. Werner-Allen, P. Swieskowski, and M. Welsh, "Motelab: a wireless sensor network testbed," in *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pp. 483–488, April 2005.

[127] X. Ju, H. Zhang, and D. Sakamuri, "Neteye: A user-centered wireless sensor network testbed for high-fidelity, robust experimentation," *Int. J. Commun. Syst.*, vol. 25, pp. 1213–1229, Sept. 2012.

[128] Rutgers University, "ORBIT – Open-Access Research Testbed for Next-Generation Wireless Networks." http://www.orbit-lab.org, 2017.

[129] University of South California, "Tutornet: A Low Power Wireless IoT Testbed." http://anrg.usc.edu/www/tutornet/, 2017.

[130] V. Handziski, A. Köpke, A. Willig, and A. Wolisz, "Twist: A scalable and reconfigurable testbed for wireless indoor experiments with sensor networks," in *Proceedings of the 2Nd International Workshop on Multi-hop Ad Hoc Networks: From Theory to Reality*, REALMAN '06, (New York, NY, USA), pp. 63–70, ACM, 2006.

[131] H. Soroush, N. Banerjee, M. Corner, B. Levine, and B. Lynn, "A retrospective look at the umass dome mobile testbed," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 15, pp. 2–15, Mar. 2012.

[132] N. Anand, E. Aryafar, and E. W. Knightly, "Warplab: A flexible framework for rapid physical layer design," in *Proceedings of the 2010 ACM Workshop on Wireless of the Students, by the Students, for the Students*, S3 '10, (New York, NY, USA), pp. 53–56, ACM, 2010.

[133] I. Chatzigiannakis, S. Fischer, C. Koninis, G. Mylonas, and D. Pfisterer, *Sensor Applications, Experimentation, and Logistics: First International Conference, SENSAPPEAL 2009, Athens, Greece, September 25, 2009, Revised Selected Papers*, ch. WISEBED: An Open Large-Scale Wireless Sensor Network Testbed, pp. 68–87. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[134] C. Cano, A. Pittolo, D. Malone, L. Lampe, A. M. Tonello, and A. G. Dabak, "State of the art in power line communications: From the applications to the medium," *IEEE Journal on Selected Areas in Communications*, vol. 34, pp. 1935–1952, July 2016.

[135] O. Hahm, E. Baccelli, H. Petersen, and N. Tsiftes, "Operating systems for low-end devices in the internet of things: A survey," *IEEE Internet of Things Journal*, vol. 3, pp. 720–734, Oct 2016.

[136] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "Tinyos: An operating system for sensor networks," in *in Ambient Intelligence*, Springer Verlag, 2004.

[137] A. Dunkels, O. Schmidt, T. Voigt, and M. Ali, "Protothreads: Simplifying Event-driven Programming of Memory-constrained Embedded Systems," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 29–42, 2006.

[138] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN)*, pp. 455–462, 2004.

[139] "CST Group at FU Berlin. Scatter web Embedded Sensor Board." http://www.csc.kth.se/~ronniej/project/Scatterweb/ESB.html.

[140] H. Will, K. Schleiser, and J. Schiller, "A real-time kernel for wireless sensor networks employed in rescue scenarios," in *2009 IEEE 34th Conference on Local Computer Networks*, pp. 834–841, 2009.

[141] "IPv6 over the TSCH mode of IEEE 802.15.4e." https://datatracker.ietf.org/wg/6tisch.

[142] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *International Conference on Local Computer Networks (LCN)*, pp. 455–462, IEEE, Nov 2004.

[143] J. Eriksson, F. Österlind, N. Finne, N. Tsiftes, A. Dunkels, T. Voigt, R. Sauter, and P. J. Marrón, "Cooja/mspsim: Interoperability testing for wireless sensor networks," in *International Conference on Simulation Tools and Techniques (Simutools)*, pp. 27:1–27:7, 2009.

[144] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: Accurate and scalable simulation of entire tinyos applications," in *International Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 126–137, ACM, 2003.

[145] A. Fraboulet, G. Chelius, and E. Fleury, "Worldsens: Development and prototyping tools for application specific wireless sensors networks," in *2007 6th International Symposium on Information Processing in Sensor Networks*, pp. 176–185, April 2007.

[146] G. Z. Papadopoulos, A. Gallais, G. Schreiner, E. Jou, and T. Noel, "Thorough IoT testbed Characterization: from Proof-of-concept to Repeatable Experimentations," *Elsevier Computer Networks*, vol. 119, pp. 86–101, 2017.

[147] B. Raman, K. Chebrolu, D. Gokhale, and S. Sen, "On the feasibility of the link abstraction in wireless mesh networks," *IEEE/ACM Trans. Netw.*, vol. 17, pp. 528–541, Apr. 2009.

[148] "A community resource for archiving wireless data at dartmouth." http://crawdad.org.

[149] G. Coulson, B. Porter, I. Chatzigiannakis, C. Koninis, S. Fischer, D. Pfisterer, D. Bimschas, T. Braun, P. Hurni, M. Anwander, G. Wagenknecht, S. P. Fekete, A. Kröller, and T. Baumgartner, "Flexible experimentation in wireless sensor networks," *Commun. ACM*, vol. 55, pp. 82–90, Jan. 2012.

[150] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet," *SIGARCH Comput. Archit. News*, vol. 30, pp. 96–107, Oct. 2002.

[151] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "Sensorscope: Application-specific sensor network for environmental monitoring," *ACM Trans. Sen. Netw.*, vol. 6, pp. 17:1–17:32, Mar. 2010.

[152] R. T. Hermeto, A. Gallais, K. Van Laerhoven, and F. Theoleyre, "Passive and Stable Initial Preferred Parent Selection through Neighbors Ranking for a Fast Convergence in 6TiSCH Networks," in *European Wireless Sensor Networks (EWSN)*, ACM, 2018.

[153] D. Cavin, Y. Sasson, and A. Schiper, "On the accuracy of manet simulators," in *Proceedings of the Second ACM International Workshop on Principles of Mobile Computing*, POMC '02, (New York, NY, USA), pp. 38–43, ACM, 2002.

[154] P. H. Gomes, Y. Chen, T. Watteyne, and B. Krishnamachari, "Insights into frequency diversity from measurements on an indoor low power wireless network testbed," in *2016 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, Dec 2016.

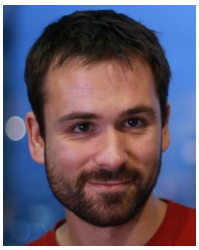
[155] A. Brandt, J. Buron, and G. Porcu, "Home automation routing requirements in low-power and lossy networks," RFC 5826, IETF, 2010.

[156] K. Pister, P. Thubert, S. Dwars, and T. Phinney, "Industrial routing requirements in low-power and lossy networks," RFC 5673, IETF, 2009.

[157] "Spectrum Requirements for Short Range Device, Metropolitan Mesh Machine Networks (M3N) and Smart Metering (SM) applications," *ETSI TC ERM, TR 103 055, v1.1.1*, Sept. 2011.

Kosmas Kritsis is a Research Associate at the Institute for Language and Speech Processing, Athena Research and Innovation Center (ILSP/ATHENA RC). He is also a Ph.D. Candidate at the Dept. of Informatics, University of Piraeus (UniPi). He holds a M.Sc. in Sound & Music Computing from Pompeu Fabra University of Barcelona (UPF) and a B.Sc. in Informatics from the Alexander Technological Educational Institute of Thessaloniki (ATEITHE). During his studies he was admitted twice the Erasmus EU scholarship for studying at the Computer Science Dept. of Carlos III University of Madrid (UC3M). Additionally, he received the Erasmus Placement EU scholarship as a student researcher at the ICube lab, University of Strasbourg (UNISTRA). His academic experience and research interest includes diverse scientific areas, such as computer networks, signal processing, computer vision, Human-Computer Interaction and machine learning.

Georgios Z. Papadopoulos (S10-M16) serves as an Associate Professor at the IMT Atlantique in Rennes, France. Previously, he was a Postdoctoral Researcher at the University of Bristol. He received his Ph.D. from University of Strasbourg, in 2015 with honors, his M.Sc. in Telematics Engineering from University Carlos III of Madrid in 2012 and his B.Sc. in Informatics from Alexander T.E.I. of Thessaloniki in 2011. Dr. Papadopoulos has participated in various international and national (FP7 RERUM, FIT Equipex) research projects. Moreover, he has received the prestigious French national ANR JCJC grant for young researchers. He has been involved in the organization of many interna- tional events (AdHoc-Now18, IEEE CSCN18, IEEE ISCC17). His research interests include Industrial IoT, 6TiSCH, LPWAN, Battery Management System and Smart Grid. Dr. Papadopoulos has received the Best Ph.D. Thesis Award granted by the University of Strasbourg and he was a recipient of two Best Paper Awards (IFIP Med-Hoc-Net14 and IEEE SENSORS14).



Antoine Gallais is an Associate Professor at the University of Strasbourg since 2008 (ICube laboratory). He holds a master (2004), a PhD degree (2007) in computer science from the University of Lille, and an HDR thesis (2017) from Univ. Strasbourg. Since sept. 2017, he is a visiting researcher at Inria Lille - Nord Europe. His research topics include wireless sensor and mobile ad hoc networking, mobility management and performance evaluation. He is serving as TPC member for several events (e.g., IEEE COMNETSAT, IEEE ICNC, IEEE Globecom), was program co-chair of ICST Adhocnets'14-15 and local co-chair of IEEE Wimob'13. He regularly serves as an external reviewer for several international journals an conferences and was an active member of several national and international research projects (ANR TLCOM SensLAB, ANR FIT Equipex, ANR INFRA IRIS, PHC EXPRESS).

Periklis Chatzimisios (S02-M05-SM12) received the B.Sc. degree in informatics from Alexander Technological Educational Institute of Thessaloniki (ATEITHE), Thessaloniki, Greece, in 2000, and the Ph.D. degree in wireless communications from Bournemouth University, Poole, U.K., in 2005. He serves as an Associate Professor and the Director of the Computing Systems, Security and Networks (CSSN) Research Laboratory in the Department of Informatics, ATEITHE. He has edited/authored eight books and more than 130 peer-reviewed papers and book chapters. His published research work has received more than 2800 citations by other researchers. His research interests include performance evaluation and standardization activities of mobile/wireless communications, Internet of Things, and big data. Dr. Chatzimisios is involved in several standardization and IEEE activities serving as a member of the Standards Development Board for the IEEE Communication Society (ComSoc), the IEEE ComSoc Standards Program Development Board, and the IEEE ComSoc Education and Training Board, as well as the Vice Chair of the IEEE ComSoc Technical Committees on Big Data (TCBD) and Information Infrastructure and Networking (TCIIN).

Fabrice Tholeyre (S05-M09-SM16) is a researcher at the CNRS. After having spent 2 years in the Grenoble Informatics Laboratory (France), he is part of the ICube lab (Strasbourg, France) since 2009. He received his PhD in computer science from INSA, Lyon (France) in 2006. He was a visiting scholar at the University of Waterloo (Canada) in 2006, and a visiting researcher at INRIA Sophia Antipolis (France) in 2005. He serves in about ten TPC per year, and is area editor for Ad Hoc Networks since 2018. He has been associate editor for IEEE Communications Letters and guest editor for Computer Communications and Eurasip JWCN. His research interests mainly concern distributed algorithms and experimental design for the Internet of Things.