A Simulation Platform for Large-Scale Internet of Things Scenarios in Urban Environments

Giacomo Brambilla
Department of Information Engineering
University of Parma
Italy
giacomo.brambilla@studenti.unipr.it

Marco Picone
Department of Information Engineering
University of Parma
Italy
marco.picone@unipr.it

Simone Cirani
Department of Information Engineering
University of Parma
Italy
simone.cirani@unipr.it

Michele Amoretti
Department of Information Engineering
University of Parma
Italy
michele.amoretti@unipr.it

Francesco Zanichelli
Department of Information Engineering
University of Parma
Italy
francesco.zanichelli@unipr.it

ABSTRACT

The Internet of Things (IoT) refers to the interconnection of billions of IP-enabled devices, denoted as "smart objects", with limited capabilities, in terms of computational power and memory capacity, which typically operate in constrained environments, in an Internet-like structure. Large-scale systems and applications that rely on such a high number of devices, due to their complexity, need careful analysis and test, before being deployed to target environments. Traditional IoT simulators do not focus on the simulation of large scale deployments, as they are intended to evaluate and analyze low-level networking aspects, with groups of smart objects arranged in specific topologies. In this paper, we illustrate an efficient simulation methodology, which is particularly suitable to test IoT systems with a large number of interconnected devices in Urban environments from an application-layer perspective. The main advantages of such an approach are: i) the capability to simulate large-scale systems with thousands of geographically distributed devices; ii) the maximization of code reuse; and iii) the high generality of simulated nodes, which can be characterized by multiple network interfaces and protocols, as well as different mobility, network, and energy consumption models.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; C.4 [**Performance of Systems**]; I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Discrete event*

Urb-IoT 2014, October 27-28, Rome, Italy

DOI 10.4108/icst.urb-iot.2014.257268

Keywords

Internet of Things, Discrete Event Simulation, Urban Environments, Smart Cities

1. INTRODUCTION

It is an established fact that in the immediate future there will be billions of objects with the capability to communicate and sense or interact with their internal states or the external environment [12]. The Internet of Things (IoT) mainly refers to the interconnection of a multitude of constrained devices with limited computational and memory capacity, typically battery-powered. As a consequence, energy efficient technologies must be adopted. Moreover, these devices usually operate in constrained networks, which often have high packet error rates and a throughput of tens of kbit/s.

In this context, standardization organisms, such as the Internet Engineering Task Force (IETF) and IPSO Alliance, have been working on standard and interoperable communication mechanisms, in order to interconnect these devices. In particular, the Internet Protocol version 6 (IPv6) [7] has been identified as the main candidate, and several working groups have been set in order to address the issues related to IoT devices communication. For example, the IETF IPv6 over Low power WPAN (6LoWPAN) Working Group [15] is defining encapsulation and other adaptation mechanisms in order to send and receive IPv6 packets over Low power Wireless Personal Area Networks, such as those based on IEEE 802.15.4. From the point of view of the application layer, the IETF Constrained RESTful Environments (CoRE) Working Group [14] is currently defining a Constrained Application Protocol (CoAP) [23], a software protocol designed to easily translate to HTTP for simplified integration with the web, while also meeting specialized requirements such as multicast support, very low overhead, and simplicity, intended to be used in resource-constrained Internet devices, such as wireless sensor network nodes.

IoT applications require a huge number of networked devices, equipped with sensors, actuators, computational and storage facilities, to cooperate and be coordinated. Due to

their complexity, such applications need careful analysis and testing, before being deployed to target environments. However, testing them in a controlled environment, such as a laboratory, may not be sufficient to understand and evaluate the possible properties/issues of such complex systems and, in particular, those that cannot be inferred from the analysis of single components, and appear when components interact. Moreover, the simulation of large-scale systems is usually based on approximated models and simulation-specific code, which are not representative of all system details.

In this paper, we propose a novel Java-based simulation platform, which allows to simulate interconnected IoT devices easily by providing an intuitive simulation methodology, which results in maximal code reuse. The proposed platform provides a general-purpose simulation engine, which includes specific packages to simulate mobility, networking, and energy consumption models. The proposed methodology allows to define general-purpose devices, which can be characterized by multiple network interfaces and protocols, as well as different network and energy models. With this solution, we can easily tackle problems like flexibility, modularity, code reuse and ease of deployment.

The remainder of this paper is organized as follows. In Section 2, we discuss relevant simulation approaches for IoT systems and applications. In Section 3, we present the architecture and the proposed methodology of our simulation platform. Section 4 describes a challenging Urban IoT case study that we have implemented to evaluate the proposed methodology. In Section 5, we draw our conclusions and discuss future work.

2. RELATED WORK

Accurate simulation of sensors taking part in wireless sensor networks is often coupled with the operating system running on top of the sensor. Most of the specialized IoT operating systems typically provide simulation environments for developers: for instance, Cooja [22] and TOSSIM [17] are the simulation platforms for testing applications running on Contiki OS [9] and TinyOS [26], respectively. The approaches of both simulators totally differ from generic simulation frameworks like OMNeT++ [27] or ns-3 [20], since they simulate the entire behavior of the node, from hardware and communication to the whole software stack running on the operating system, which precludes the possibility to simulate a high number of deployed nodes.

These platforms are targeted at evaluating specific aspects (*e.g.*, link- or application-layer protocols and energy consumption), other frameworks, such as ns-3, provide a much wider variety of network and communication protocols and Internet system representations. However, the entire domain of wireless sensor networks has been affected by the increasing importance of the Internet of Things and related technologies, such as 6LowPAN [15]. As standardization efforts are being carried out, resulting in the design of standard communication protocols on top of IP, developers are starting to out the ever-increasing need for simulation platforms, which can be used to test large-scale systems comprising a high number of nodes, without taking into account low-level issues (*i.e.*, hardware platforms and operating systems), but focusing on application-specific issues. Weingärtner *et al.* in [28] report the need for an accurate IoT simulator and discuss different approaches to overcome the limitations of available simulators. Among all the approaches, a hybrid simulation environment, based on a combination of available generic frameworks and system-level simulators together, might reduce the gap between research and practical deployments and strengthen the option of simulative studies for the IoT area.

In [16], the author highlights the need for interconnection between the two different types of network simulators and suggests the integration of an accurate operating system simulator into a generic simulation environment with a correct representation of protocols from the different layers of the used network stacks. The MAMMoTH project aims at building an emulation platform that can support nodes in the order of tens of millions [18], but it has not been recently updated.

3. PROPOSED METHODOLOGY

The proposed methodology is based on the concept of *IoT Node*, which represents a generic smart object endowed with a mobility model, one or more network models, and an energy model, which can interact and provide feedback among themselves to better characterize the behavior of simulated nodes. The mobility model defines the movement of the IoT Node, and how its location, velocity, and acceleration change over time. Network models describe network capabilities of each interface the IoT Node is equipped with. For example, these models define delays and failure rates in the delivery of data in the network. Finally, the energy model describes the behavior of the IoT Node from an energy consumption point of view and can also take into account duty-cycling.

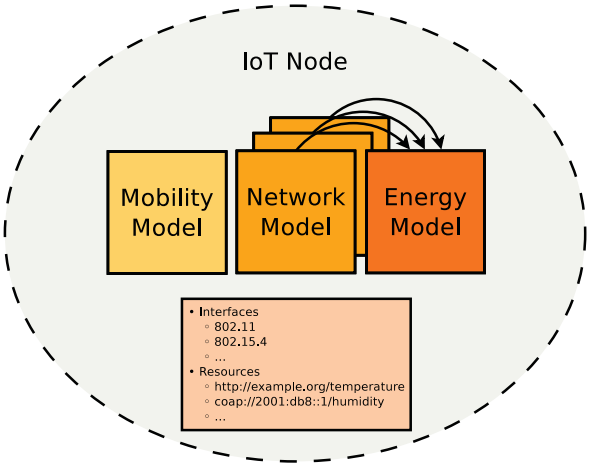

Figure 1: Visual model of an IoT Node.

A visual model of the IoT Node is shown in Figure 1. Network models are linked to the energy model, since the use of network interfaces affects energy consumption, which varies depending on the type of access network technology.

An IoT Node is also characterized by i) different network interfaces (*e.g.*, IEEE 802.15.4, IEEE 802.11, IEEE 802.3, or Bluetooth), ii) various communication application-layer protocols (either standard, such as CoAP [23], HTTP [10] or MQTT-SN [25], or user-defined, such as CoSIP [5]); and iii) resources, which may be identified by their Uniform Resource Identifier (URI) [3].

Through such a definition of IoT Node, our simulation platform reveals a high flexibility: it is possible to simulate different applications which make use of IoT devices characterized by any mobility, network and energy models. Furthermore, being composed of models with a loose coupling, our IoT Node has a high modularity and allows the reuse of most of the code. The replacement of the energy model does not require to modify the others, for example. Moreover, by adopting the required interfaces, it is possible to directly deploy the source code on real devices.

3.1 Architecture

Starting from the concept of IoT Node, the main class of our simulation platform: the `IoTNode` class has been defined. By extending this class, developers can implement and simulate IoT applications. The extension of generic IoT Nodes allows to isolate the application layer from the underlying layers and brings the advantage to focus on application-specific aspects and, reuse the entire application logic (and implementation, where applicable).

The architecture of the proposed simulation platform is shown in Figure 2. The architecture is modular and based on several technologies. The top layer of the architecture, depicted in Figure 2, represents the application to be tested, whereas the underlying layer (denoted as *Adaptation Layer*) is intended as the coordination of the `IoTNode`s. The remainder of the architecture and the main features of core components of our simulation framework are detailed next.

3.2 DEUS

Since the objective is the simulation of a high number of IoT Nodes, we searched for a generic simulation engine and the choice fell on DEUS [1] for its high scalability and versatility. DEUS is a general-purpose discrete event simulation environment. It is a free software project developed in Java. Its APIs allow developers to implement (by sub-classing) (i) *nodes*, *i.e.*, the entities which interact in a complex systems, leading to emergent behaviors; (ii) *events*, *e.g.*, node births and deaths, interactions among nodes, interactions with the environment, logs and so on; and (iii) *processes*, either stochastic or deterministic ones, constraining the timeliness of events.

Furthermore, it exists an extension of DEUS dedicated to the modeling of mobile nodes that maintains its characteristics of generality and abstraction. Such an extension, called OSMobility, allows to easily realize and model the aspects related to mobility of IoT Nodes [4].

3.3 OSMobility package

OSMobility [8] is a simulation environment which allows to simulate the motion of different entities, such as pedestrians and vehicles, in realistic geographical spaces. Being based on DEUS, OSMobility inherits all the features that make it versatile and generic.

The main class of OSMobility is `GeoNode`, which is a generic element of a simulation, and is characterized by a geographical location. It is extended by `StationaryNode` and `MobileNode`, which represent a static node (*e.g.*, a traffic light, roadside sensor) and a mobile node, respectively. OSMobility is integrated with OpenStreetMap (OSM) [24], an open database which provides geographical data, such as road maps, for free, and uses such data to compute node trajectories, also taking into account speed limits and points of interest. Thus, OSMobility allows to simulate vehicles running on highways or urban roads, pedestrian walking within Limited Traffic Zones, bicycles moving on cycling lanes, etc.

One of the main features of OSMobility is the ability to define any mobility model, using the abstract class `MobilityModel` as a common base. Such a class has been specialized to `FluidTrafficModel`, which implements the Fluid Traffic Model (FTM) [10], where speed is a monotonic decreasing function of nodes density. The FTM is used to compute the next position of a mobile node, given its current position, speed, direction, and the density of surrounding nodes. Such a computation is performed by the `MoveNodeEvent`, whose instances are continuosly generated and inserted in the event queue of the simulation engine, for every `MobileNode` that is moving.

By adopting the Adapter pattern [11], we have designed our `IoTNode` as a wrapper of OSMobility's `GeoNode`. Thus, it is a node of the simulation and provided with a geographical location. It can be whether a stationary node or a mobile node. In the latter case, it can adopt any mobility model, keeping the high flexibility of OSMobility.

3.4 Communication and Energy packages

An `IoTNode`, in addition to a mobility model, has a network model and an energy model, represented by the generic classes `NetworkModel` and `EnergyModel`, respectively. By extending the `NetworkModel` class, it is possibile to describe the capabilities of the node with regard to the network interfaces and protocols of communication. Indeed, developers can customize an `IoTNode` in order that it represents the required IoT device, with any network interface and protocol of communication. The `NetworkModel` has the task to calculate delays or even failures in the delivery procedure.

To simulate a distributed system with DEUS, it is necessary to write the classes that represent a message delivery from one node to another. In particular, it is necessary to define the sender, the destination and to schedule a *delivered message event* in the future (in terms of virtual time of the simulation). The scheduling time of such an event must be set using a suitable process, selected among those that are provided by the DEUS API, or defined by the user, possibly. The communication package provides several delay models which can be used to simulate message transmission between network nodes. The most simple model generates an exponential delay, whose expected value is computed from the message size and the nominal channel bandwidth. If the purpose of the simulation is to measure the average delay of propagating multi-hop messages within a network of nodes, the value of each link's delay must be realistic, taking into account the underlying networking infrastructure. In particular, if the communication is wireless, estimating the delay of point-to-point communication is a challenging task. Fortunately, this is not necessary, thanks to possibility to use dedicated simulation tools, such as Cooja and ns-3, to fully characterize the communication delay and packet losses.

Furthermore, the developer can implement the desired energy model, extending the `EnergyModel` class, so that it is possible to model in a highly sophisticated way, the battery consumption of the device. For example, it is possible to model the rate of the duty cycle for each `IoTNode`. Also in this case, Cooja can be used to obtain a very detailed modeling of the energy consumption.

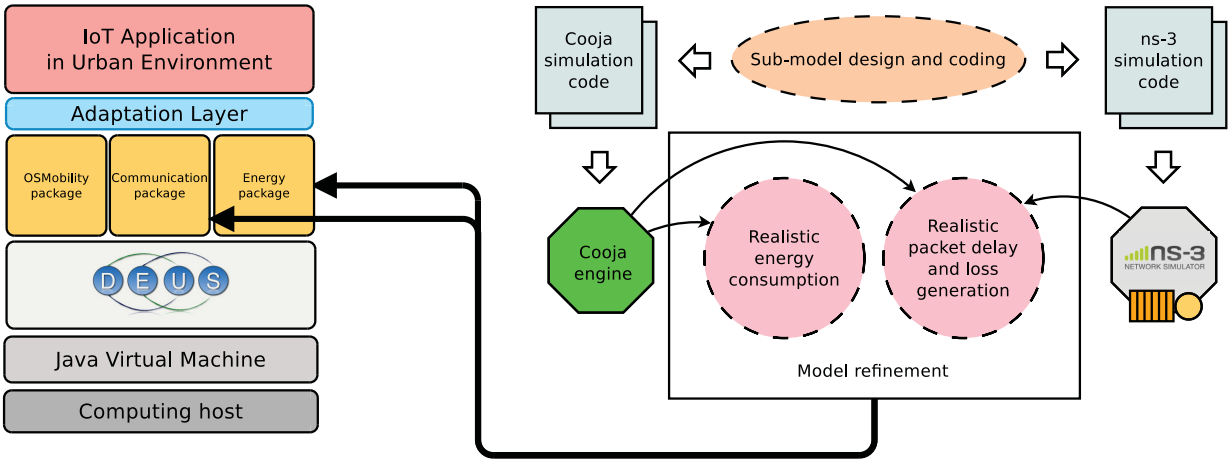

Figure 2: Layered representation of the proposed methodology.

3.4.1 *Cooja and ns-3*

Cooja is a network simulator included in the Contiki system, which is an open source operating system for networked, memory-constrained systems with a particular focus on low-power wireless IoT devices. Contiki is often used in street lighting systems, sound monitoring for smart cities, radiation monitoring systems, and alarm systems. Cooja simulates networks of Contiki nodes, which may belong to either of three classes: emulated nodes, where the entire hardware of each node is emulated, Cooja nodes, where the Contiki code for the node is compiled for and executed on the simulation host, or Java nodes, where the behavior of the node must be reimplemented as a Java class.

Ns-3 is a discrete event network simulator for Internet systems. It is a free software project publicly available under the GNU GPLv2 license for research, development, and use. The ns-3 project is committed to building a solid simulation core that is well documented, easy to use and debug, and that caters to the needs of the entire simulation workflow, from simulation configuration to trace collection and analysis. Furthermore, the ns-3 software infrastructure encourages the development of simulation models which are sufficiently realistic to allow ns-3 to be used as a real-time network emulator, interconnected with the real world and which allows many existing real-world protocol implementations to be reused within ns-3. The ns-3 simulation core supports research on both IP and non-IP based networks. However, the large majority of its users focuses on wireless/IP simulations which involve models for Wi-Fi, WiMAX, or LTE for layers 1 and 2, and a variety of static or dynamic routing protocols such as Optimized Link State Routing Protocol (OLSR) and Ad hoc On-Demand Distance Vector (AODV) for IP-based applications.

3.4.2 *Model refinement*

The direct integration of DEUS with Cooja and ns-3, with the first that "calls" the others to compute a delay value every time a node must send a message to another node and the energy consumption, taking into account current surrounding conditions, is unpractical and would highly increase the simulation time. Instead, a more effective and efficient solution includes the following steps:

1. identify the main sub-system types, each one being characterized by specific networking features;

2. with Cooja or ns-3: create detailed simulation models of the sub-systems (*i.e.*, sub-models), and measure their characteristic transmission delays and the power profiling;

3. with DEUS: simulate the whole distributed system, with refined scheduling of communication events, taking into account the transmission delays and the realistic power consumption computed at step 2.

Amoretti *et al.* in [2] give a more detailed explanation of the model refinement process from ns-3. A similar approach can be also adopted for Cooja.

4. CASE STUDY

To evaluate our simulation platform and, in particular, its scalability, we have decided to define several simulation scenarios and, for each of them, we have calculated the required time to complete the simulation.

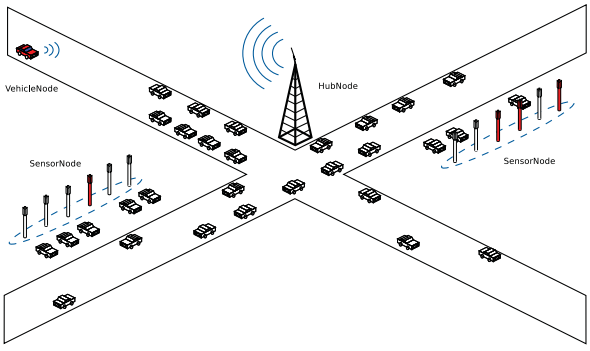

Figure 3: Sample IoT (smart-parking) scenario for the simulations.

The scenarios we have decided to simulate belong to a typical application case of the Internet of Things in urban environments: a smart parking infrastructure where in each parking lot a sensor node is deployed to detect the presence or absence of a vehicle [13]. Moreover, there are gateways that take charge of collecting data from the sensor network, and vehicles that move around the city and contact the gateways, searching for an available parking lot.

In particular, we have defined that vehicle requests to gateways are scheduled by an Homogeneous Poisson Process with interarrival time being an exponentially distributed random variable whose mean interarrival time is equal to 5 minutes. When a vehicle makes a request, it communicates with the geographically nearest gateway. The number of vehicles looking for a parking lot is chosen so that, at the end of the simulation, 10% of the traveling vehicles have issued a parking request to the gateways. Next, the gateway contacts the parking sensors under its own responsibility and waits for their response. Therafter, it communicates a response to the vehicle. All this happens for a lifetime of the simulation equals to 4 hours. Figure 3 shows our simulation scenario, where some vehicles travel around a generic city, send messages with gateways using LTE communication standard and also presents wireless sensors networks which transmit their data to the gateways.

All the participants of the simulation are implemented by subclassing our implementation of the `IoTNode` class. In particular, they are described by the `VehicleNode`, `HubNode` and `SensorNode` classes. In addition, they adopt the interface of mjCoAP [6], a well-known open-source Java implementation of the CoAP protocol. In this way, code portability is even higher and the implementation can easily be transferred on real devices, such as, for example, those equipped with the new Java ME 8 platform [19].

In order to assess the scalability of our simulation platform, the number of nodes of the simulation has been varied, as reported in Table 1, and the required computation time has been measured. Simulations have been executed on a server with 2 GHz Xeon CPU, 16 GB RAM, Ubuntu GNU/Linux operating system.

	# `SensorNode`	# `HubNode`	# `VehicleNode`
A	4000	8	500
B	8000	16	1000
C	20000	40	2500
D	40000	80	5000
E	100000	200	12500
F	200000	400	25000

Table 1: Number of nodes of each simulation scenario.

Since the complexity of a simulation in a discrete-event simulation framework depends mostly on the number of events, rather than the number of nodes, for each simulation scenario, the number of scheduled events has been calculated, and reported in Table 2. It also presents also the percentage of mobility and communication events with respect to the total number of scheduled events. In particular, in our simulations we have adopted a very accurate mobility model and this entails the prevalence of mobility events rather than communication or birth events.

	# events	% mobility	% communication
A	2.7×10^6	98.5	1.4
B	5.6×10^6	98.5	1.3
C	1.4×10^7	98.5	1.3
D	2.7×10^7	98.4	1.4
E	6.3×10^7	98.3	1.5
F	1.2×10^8	98.2	1.6

Table 2: Number of events of each simulation scenario.

Figure 4 shows the results of simulations in terms of execution time. As the reader can see, the presented simulation platform can easily manage a very high number of nodes and the associated events. Considering that the SmartSantander testbed [21], which is one of the most famous real IoT platforms in urban environment, is composed of about 3000 IEEE 802.15.4 devices, 200 GPRS modules and 2000 joint RFID tag/QR code labels deployed both at static locations (streetlights, facades, bus stops) as well as on-board of mobile vehicles (buses, taxis), we can state that our simulation platform allows to easily simulate realistic scenarios in short time (scenarios B and C). Moreover, even with the enormous number of nodes we have considered in scenario F, the computation time is still acceptable, making the simulation platform particularly suitable for urban environment scenarios, where the number of nodes is often considerable.

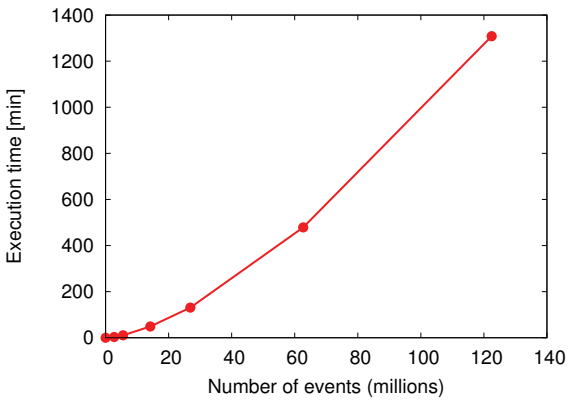

Figure 4: Execution time with respect to the number of events, for the six considered scenarios.

5. CONCLUSION

In this paper we have proposed a simulation platform, that is particularly suitable for testing large-scale systems and applications for the Internet of Things in urban environments. Since it is based on a general-purpose simulation engine and a set of modular simulation packages (mobility, communication and energy consumption packages), the only effort that is required is the integration of deployment code with the simulation packages, by extending the `IoTNode` class. We have illustrated the organization and current implementation of the proposed platform and modeled a characteristic

case of IoT application in an urban environment, for a preliminary assessment of its scalability.

Regarding future work, we plan to define more detailed simulation models to study scenarios with a very high heterogeneity of devices, and to perform even more realistic large-scale simulations of IoT systems in urban environments.

6. REFERENCES

[1] M. Amoretti, M. Agosti, and F. Zanichelli. DEUS: A Discrete Event Universal Simulator. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, Simutools '09, pages 58:1–58:9, ICST, Brussels, Belgium, Belgium, 2009.

[2] M. Amoretti, M. Picone, F. Zanichelli, and G. Ferrari. Simulating mobile and distributed systems with deus and ns-3. In *High Performance Computing and Simulation (HPCS), 2013 International Conference on*, pages 107–114, July 2013.

[3] C. Bormann, A. Castellani, and Z. Shelby. CoAP: An application protocol for billions of tiny internet nodes. *IEEE Internet Computing*, 16(2):62–67, 2012.

[4] G. Brambilla, A. Grazioli, M. Picone, F. Zanichelli, and M. Amoretti. A cost-effective approach to software-in-the-loop simulation of pervasive systems and applications. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, pages 207–210, March 2014.

[5] S. Cirani, M. Picone, and L. Veltri. CoSIP: A Constrained Session Initiation Protocol for the Internet of Things. In C. Canal and M. Villari, editors, *Advances in Service-Oriented and Cloud Computing*, volume 393 of *Communications in Computer and Information Science*, pages 13–24. Springer Berlin Heidelberg, 2013.

[6] S. Cirani, M. Picone, and L. Veltri. mjCoAP: An Open-Source Lightweight Java CoAP Library for Internet of Things applications. In *22nd International Conference on Software, Telecommunications and Computer Networks - (SoftCOM 2014), Workshop on Interoperability and Open-Source Solutions for the Internet of Things*, September 2014.

[7] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, December 1998.

[8] Distributed Systems Group. OSMobility - Discrete Event Simulator for Vehicular Mobility. `https://code.google.com/p/osmobility/`.

[9] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462, Nov 2004.

[10] R. T. Fielding, J. Gettys, J. C. Mogul, H. F. Nielsen, L. Masinter, P. J. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, June 1999.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*.

[12] Gartner Inc. Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020. `https://www.gartner.com/newsroom/id/2636073`, December 2013.

[13] A. Grazioli, M. Picone, F. Zanichelli, and M. Amoretti. Collaborative mobile application and advanced services for smart parking. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, volume 2, pages 39–44, June 2013.

[14] IETF. IETF Constrained RESTful Environments Working Group. `http://tools.ietf.org/wg/core`.

[15] IETF. IETF IPv6 over Low Power WPAN Working Group. `http://tools.ietf.org/wg/6lowpan`.

[16] M. Kirsche. Simulating the Internet of Things in a Hybrid Way. In *Proceedings of the Networked Systems (NetSys) 2013 PhD Forum*, March 2013. Poster Abstract.

[17] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *SenSys'03: Proceedings of the First International Conference on Embedded Networked Sensor Systems*, pages 126–137, 2003. cited By (since 1996)395.

[18] V. Looga, Z. Ou, Y. Deng, and A. Yla-Jaaski. Mammoth: A massive-scale emulation platform for internet of things. In *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*, volume 03, pages 1235–1239, Oct 2012.

[19] S. Meloan and T. Barr. Java ME 8 and the Internet of Things. `http://www.oracle.com/technetwork/articles/java/ma14-java-me-embedded-2177659.html`.

[20] ns-3 project. ns-3 - Discrete Event Network Simulator. `http://www.nsnam.org`.

[21] OpenStreetMap contributors. OpenStreetMap. `http://www.openstreetmap.org`.

[22] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 641–648, Nov 2006.

[23] Z. Shelby, K. Hartke, and C. Bormann. Constrained Application Protocol (CoAP). RFC 7252, June 2014.

[24] SmartSantander's Team. SmartSantander. `http://www.smartsantander.eu`.

[25] A. Stanford-Clark and A. Nipper. MQ Telemetry Transport (MQTT) Protocol. `http://mqtt.org`.

[26] TinyOS Alliance. Tinyos. `http://www.tinyos.net`.

[27] A. Varga and R. Hornig. An Overview of the OMNeT++ Simulation Environment. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Simutools '08, pages 60:1–60:10, ICST, Brussels, Belgium, Belgium, 2008.

[28] E. Weingärtner, M. Ceriotti, and K. Wehrle. How to simulate the Internet of Things? In *Proceedings of the 11th GI/ITG KuVS Fachgespraech Sensornetze (FGSN 2012)*, Sep 2012.