Chapter 4
Internet of Things Simulation Using OMNeT++ and Hardware in the Loop

Philipp Wehner and Diana Göhringer

4.1 Introduction

While the internet was primarily used by human beings a couple of years ago, a trend shows up, where "intelligent things" use the network infrastructure to communicate with each other and to exchange data. The aim is to replace the personal computer, as known in its common form, by something that supports humans by their daily activities and procedures, without getting perceived directly [1]. Problematic in this context are the different demands of individual network nodes regarding the network infrastructure. Efficient communication is affected by different protocols that are used [2]. To face this challenge, standardization can help to create preferably huge subnetworks that use a consistent procedure to communicate. But this method is only meaningful as long as it does not produce high costs for individual network participants and complexity of the standard is in a balanced relation to the obtainable benefit. This is not at least related to the energy that is consumed for the standard-compliant communication. The connection of basic sensors, e.g., window or door contact switches in the smart home, would be reduced to absurdity, when a high-cost communication medium, as, for example, Ethernet is used. High acquisition and operating cost would be the result. Hence, utilizing a common interface for communication is meaningful only when requirements of the individual network participants are compatible with the targeted standard. At this, it should get clear that value has to be attached to the interconnection of subnetworks via gateways. To enable the communication across the subnetworks, gateways must be deployed where standardization is not a means to an end. As the complexity of this consideration, especially for huge networks, is

P. Wehner (✉) • D. Göhringer
Application-Specific Multi-Core Architectures (MCA) Group, Ruhr-University Bochum, Bochum, Germany
e-mail: philipp.wehner@rub.de; diana.goehringer@rub.de

G. Keramidas et al. (eds.), *Components and Services for IoT Platforms*,
DOI 10.1007/978-3-319-42304-3_4

not negligible, new technologies have to be developed that allow the exploration of this issue. Particularly to avoid problematic cases which are difficult to investigate in a real world environment, simulation techniques need to be developed that allow a safe but accurate observation.

Subject of this paper is the introduction of a concept that starts by developing a basic gateway. Over the course of time, it can be transferred to a simulator for the Internet of Things (IoT). To achieve this, the network simulator OMNeT++ is introduced that can be utilized for the research on the communication infrastructure. But the target of the paper is not only the simulation of Ethernet networks; it rather introduces a technique to simulate different communication standards in the IoT. It further should be made clear that the presented concept is not limited to an enclosed simulation. If real hardware components, e.g., sensors and actuators are available, they can be integrated in the simulation environment. This Hardware in the Loop (HiL) can essentially increase quality and significance of the results, as the output is not based on probably inaccurate models of the periphery. Furthermore, HiL enables the analysis of components that are not available at an early stage of the development cycle. By using the proposed simulation environment, virtual devices can exist among their physical counterparts. A framework that allows the easy extensibility of IoT networks is of high importance and HiL can enable a rapid prototyping. But there are drawbacks. Especially in large networks, the synchronization between nodes is an important aspect. The presented work enables further analysis on synchronization techniques between the physical devices and the HiL environment. Metrics regarding power consumption [3] and performance [4] can also be considered and analyzed, allowing easy measurements in the entire environment.

The paper is structured in the following manner: Sect. 4.2 introduces the network simulator OMNeT++. Section 4.3 gives an overview of related work. In example of the EU project "RADIO," introduced in Sect. 4.4, the requirements of mobile robot platforms to the communication infrastructure are presented and it is shown, how to deal with these challenges. The concept of the IoT simulation can be found in Sect. 4.5. It starts with an introduction to the idea, including the consideration of HiL and is transferred to a case study, presented in Sect. 4.6. The paper concludes with Sect. 4.7.

4.2 OMNeT++

OMNeT++ allows the discrete event simulation of computer networks and other distributed systems [5]. With the target to model large networks, the C++ based project tries to fill the gap between research-oriented simulation software and expensive commercial alternatives, such as OPNET, which is now part of Riverbed Technology [6]. OMNeT++ allows the hierarchical organization of simulation models. The number of layers is hereby not limited. This modular structure supports clarity and facilitates the work with the simulation system. Processes within the

virtual network can be visualized via a graphical user interface. This includes the illustration of network graphics, data flow charts, and the possibility of observing objects and variables during simulation. In OMNeT++, the structure of the model is defined using network description (NED) files. Here, parameters are defined for the individual modules and also their connection to other modules and submodules is specified. To efficiently manage large networks, definitions can be split into separate NED files. It is also possible to define parameters for the respective topologies. In this point, OMNeT++ is distinguished from other graphical editors, as it is not limited to fixed topologies. The topology is rather modifiable even during runtime. The user optionally can edit the NED files either via a graphical user interface or based on text files. OMNeT++ includes a library that can be used for the C++ based programming of the modules. It has to be differentiated between two possible implementations: On the one hand, *co-routine based programming* generates a thread for each module where the program code is executed. The respective thread gets the control from the simulation kernel, when a message arrives. In this case, a thread usually never returns; an infinite loop within the module continuously handles incoming and outgoing messages. On the other side, *event-processing* calls the module-function with the message as a parameter. Here, the function immediately returns after processing. Event-processing is feasible especially for extensive simulations, as not every module needs to have its own CPU stack and memory requirements are reduced. In the context of this paper, OMNeT++ is used, as it has the following benefits compared to the related work:

- OMNeT++ is extensible. C++ program code can be used to describe functionality of the respective modules. Furthermore it is possible to integrate OMNeT++ in larger applications by replacing its user interfaces.
- OMNeT++ already contains protocol models like TCP/IP, ATM, and Ethernet.
- The graphical user interface *Tkenv* supports modelling of complex networks.
- OMNeT++ was developed for the simulation of extensive networks.
- As OMNeT++ is open source, functionality of every part of the simulation can be comprehended.

4.3 Related Work

The bidirectional communication within home automation networks results in high costs, since each node needs to send and receive data. The benefit of this approach is the higher reliability of the network, as each node can acknowledge packets. In [7], the authors present a technique that is based on less-expensive, unidirectional nodes. A sequence of packets is sent with a constant inter-packet time, guaranteeing that at least one transmitted packet reaches its destination node. The approach is evaluated using an OMNeT++ based simulation. It satisfactorily shows that OMNeT++ can be used for home automation related networks.

Not at least when considering healthcare under the IoT paradigm, the importance of suitable simulation capabilities becomes clear. The authors of [8] present *SimIoT*, a toolkit that can be used to enable experiments on the interactions within an IoT scenario. As monitoring health is an important aspect in the RADIO project, presented in Sect. 4.4, SimIoT is relevant in context of the OMNeT++ based approach presented in this paper.

In context of wireless sensor networks, several approaches exist to simulate these complex environments. OBIWAN [9] is one of those simulators, focusing on cross-layer dependencies and the challenges of large networks that are not sufficiently handled by many of the available simulation tools. The authors present a case study with several thousand nodes, demonstrating the usage of their simulator in large networks. OBIWAN is based on OMNeT++.

There are ambitions to create standards for the interconnection of smart home devices. When a single device is used to manage the communication within the smart home, the whole network is vulnerable. The "Thread Group," as an example, tries to overcome this issue by creating a networking protocol that encourages the development of the IoT [10]. Amongst others, the proposed standard is simple to use, power efficient, and has no single point of failure.

In this context, the Mediola gateway must be mentioned [11]. Its great advantage is the combination of different smart home products to flexibly automate the living environment. The consumer can control the different technologies using a smartphone app. However, the Mediola gateway exactly represents the single point of failure that was mentioned above.

This paper by contrast presents a framework that also provides gateway functionality, but in a decentralized manner. The simulation allows the translation between different communication technologies, but is not dependent on a single device.

A strategic research roadmap for the IoT is presented in [12]. The authors mention the challenges of numerous heterogeneous components and emphasize the need for innovative design frameworks, inspired by HiL. However, HiL is often used in the context of automotive applications [13–15], but has only minor importance in the IoT simulation.

To the best of the authors' knowledge, there is no flexible simulation environment for the IoT that combines decentralized gateway functionality with HiL, enabling the design and analyses of large networks.

4.4 Robots in Assisted Living Environments

As demography changes and the population of elderly grow, new paradigms have to be invented to face the challenges of the increasing life expectation. Dealing with chronic diseases results in the need for long term care and innovative technologies to overcome this issue. The classical institutional care hits the brick wall under these challenges as several problems occur, not at least resulting in high costs. Modern information and communication technology (ICT) offers new solution approaches.

The project "RADIO," funded by the European Union's Horizon 2020 research and innovation program, shows up a way to support elderly in their homely environment [16]. RADIO, as short for "Robots in Assisted Living Environments: Unobtrusive, efficient, reliable, and modular solutions for independent ageing" is based on the four main dimensions *user acceptance*, *integrated and power-aware data collection—transmission—processing*, *user interfaces*, and *architecture*. It is the triggering event of the approach presented in this paper and targets the integration of smart home technology and robotic in ways that the user perceives the technical equipment as a natural component, as part of the daily life. Hence, a robot gradually becomes as ordinary as, for example, a radio. To succeed in this challenge, novel approaches to unobtrusiveness and acceptance need to be developed and transferred to a system that satisfies the need for integrated smart home technology together with assistant robot platforms. Instead of utilizing discrete sensing equipment, RADIO rather uses a robot as a mobile sensor for health monitoring. This robot, however, presents special challenges to the ICT as it is mobile.

In the context of this paper, the information technology needs to be capable of handling difficulties regarding the wireless connection. Concepts must be developed that deal with disturbances and disconnection, where data gets delivered delayed and possibly from a completely different spatial location. For this purpose, it is necessary, that suitable data buffers are available and that their integration in the network can be analyzed. Hence, a framework for the simulation of IoT networks, as conceptual introduced in this paper, is beneficial insofar, as complex processes can be broken down. Concepts for problem solving can be developed and transferred to the final application.

4.5 Concept

OMNeT++ enables the simulation of large networks and allows the integration of HiL by design. The main challenge of connecting physical devices is its integration to the scheduling mechanism of the simulation environment. OMNeT++ therefore provides a real time scheduler that can be extended by the user. The socket example that comes with the installation of OMNeT++ demonstrates how external applications can get access to the simulation environment: An extension of the real time scheduler opens a TCP/IP socket listening on port 4242. The user can connect to this server with a web browser. The respective HTTP request is visualized in the GUI and shows how data is transferred via the internet to the web server and then sent back to the client. This example sufficiently demonstrates how the interaction between HiL, in form of the client requesting a website, and the scheduler is performed. It also shows that this integration uses a single entry point, in form of the extended scheduler, to connect the two environments, the physical and the virtual world. According to this, the question comes up on how multiple hardware devices can be added to the simulation at different entry points. In case of IoT devices, that could mean that different sensors and actuators need to connect to different

nodes in OMNeT++. The socket example does not provide information to face this challenge, as it only shows how a single external device, the web client, can be integrated to the simulation environment. To understand this issue it must become clear that OMNeT++ gets informed about the modified scheduler by an entry in the initialization file of the respective simulation. As this entry is a *general* item, the drawback is that this information affects the entire simulation and hence it is not possible to provide a second scheduler for a second HiL device this way. This naturally makes sense as only one scheduler can handle the simulation process. It is therefore required to develop a mechanism that allows the integration of multiple external devices, on different network nodes, using only one extended version of the scheduler. The approach presented in this paper is based on callback functions that are consecutively invoked by the scheduler. Hence, each node, connecting physical devices to the simulation environment, must provide a method that informs the scheduler about relevant events that need to be included in the simulation flow. The following subsections show how this mechanism is realized in detail.

4.5.1 Modified Scheduler

The modified scheduler, `GatewayRTScheduler`, presented in this paper, is derived from the OMNeT++ `cScheduler` class. It provides a method `setInterface()` that enables the user to add an interface to external hardware. The virtual function `getNextEvent()` of `cScheduler` is replaced by an implementation that consecutively calls these interface functions to check if an event occurs. The function `setInterface()` therefore adds all HiL interfaces to a vector, containing a reference to the calling module, a `cMessage` object used to identify the respective event and the actual callback function.

4.5.2 HiL Interfaces

The interfaces to the external hardware devices are provided in the form of a shared library. In OMNeT++, a shared library is created by invoking `opp_makemake` with a `--make-so` flag. It mainly consists of the two functions: `virtual void handleMessage(cMessage *msg)` and `static cMessage *interfaceFunction(cModule *module, cMessage *notificationMsg, simtime_t t, cSimulation *sim)`, whereas `msg` is a pointer to the message that must be processed, `module` is a pointer to the module that contains the callback function, `t` is the current simulation time, and `sim` is a pointer to the simulation manager object. `interfaceFunction()` is the callback function that is assigned to `setInterface()` of the gateway real time scheduler class. Its task is to check whether an event of the respective HiL device needs to get considered by the simulation. The task of `handleMessage()`

can be described as follows: The function is called every time a message is present at the respective module. This message can be (a) an external message coming from a distant module, (b) the so-called *self message* that is sent and received within the same module, or (c) the message that was handed over to the scheduler to inform the appropriate module about HiL events. Hence, `handleMessage()` consists of the following distinction of cases:

```
if (msg->isSelfMessage()) {
    // handle self message
} else if (msg == hilEvent) {
    // handle HiL event
} else {
    // handle message from distant module
}
```

4.5.3 Messages

To face the challenge of different communication standards within a large network, messages of network attendees are translated to an intermediate representation (IR) that is independent from the actual protocol it represents. As this IR should not add an additional payload to the communication mechanism, it must be as lean as possible. In the context of this paper, the IR consists of concatenated integer representations of all relevant input data. Figure 4.1 visualizes this behavior.

The fictitious message coming from the smart home switch consists of a number, identifying the device and its current state (on = 1/off = 0). This message could be transferred to the IR using, for example, the following XML description:

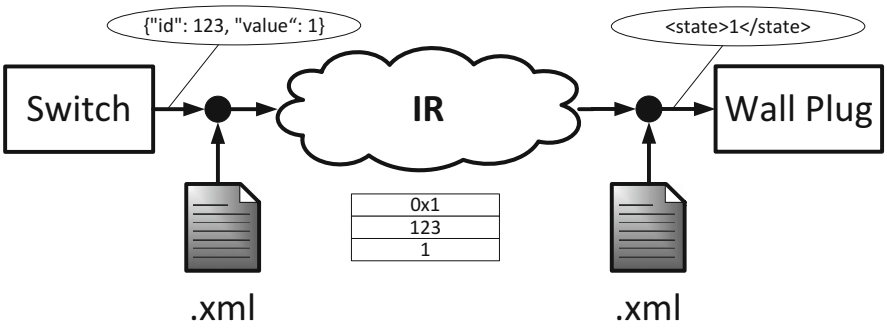

Fig. 4.1 Message translation example

```
<ir>
    <item pos='0'>
        <fixed>0x1</fixed>
    </item>
    <item pos='1'>
        <variable>
            <uint8>id</uint8>
        </variable>
    </item>
    <item pos='2'>
        <bool>
            <variable>state</variable>
        </bool>
    </item>
</ir>
```

In this case, the intermediate representation of the incoming message would consist of the values 0×1, 123, and 1, representing a unique identifier for the category of smart home switches (0×1), the id of the switch that takes action (123), and the current state (1 = on). The idea is that this IR is only based on variables that can be stored in one or more registers. Therefore, no complex datatypes, for example, "string," are allowed. The respective size that has to be reserved in the IR can be specified by the XML description that is responsible for the incoming message.

In example of Fig. 4.1, the switch should be connected to a wall plug. Considering that this actuator has a communication technique different from the switch, an additional translation mechanism must be included to transfer the messages from the IR to the actual target language. This can exemplarily be done by providing the following XML translation file:

```
<ir>
    <item name='state'>2</item>
</ir>
```

It says that the IR contains the value for the "state" variable at position 2. Using this information, the destination node can translate the information of the IR to the format that must be transferred to the wall plug interface.

The presented format of the IR was selected as OMNeT++ supports this structure in its message files. For the case study, presented in the next section, the following object is used:

```
message GatewayMsg
{
    int source;
    int destination;
    int payload[];
}
```

The integer variables `source` and `destination` contain the ids of the sender and receiver node respectively. `Payload` is an array that consists of the values of the IR.

OMNeT++ provides functions for its dynamic allocation. Adding a new entry is done as follows:

```
int oldSize = msg->getPayloadArraySize();
msg->setPayloadArraySize(oldSize+1);
msg->setPayload(oldSize, newEntry);
```

4.6 Case Study

The case study shows how HiL can be added to OMNeT++ using the modified scheduler. There are different techniques on how real hardware can interface with the simulation environment. Using vendor-specific USB dongles, e.g., for Z-Wave communication is one option to enable the data transfer between the respective node and the actual IoT simulator. For the first version of the framework presented in this paper, simple TCP/IP sockets are used as they provide a high degree of flexibility. Therefore, two Raspberry Pis, acting as interfaces, are used to manage the communication on the sender and receiver side. Each Raspberry Pi has a connection to the OMNeT++ simulation environment via Ethernet. The case study implements the concept visualized in Fig. 4.2. Therefore, an EnOcean energy harvester [17] is connected to the first Raspberry Pi using the so-called EnOcean Pi add-on module [18]. It enables the Raspberry Pi to read incoming EnOcean messages via Universal Asynchronous Receiver Transmitter (UART). The Raspberry Pi connects to the socket, opened by the EnOcean interface that was added to OMNeT++. On the receiver side, a Z-Wave module, RaZberry [19], connects the second Raspberry Pi and a Z-Wave wall plug [20]. When the EnOcean switch gets pushed, a message is transferred via Ethernet to the OMNeT++ simulation where it is wrapped into the message object presented in the previous section. The respective switch-state is added to the payload array and then sent to the receiver node, where the message is transferred to the wall plug using the second Raspberry Pi.

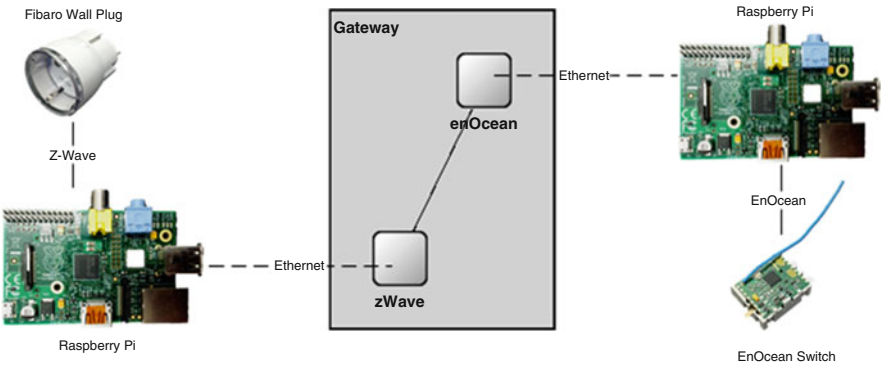

Fig. 4.2 OMNeT++ simulation acting as gateway between HiL devices

The case study currently does not include the implementation of the XML translation mechanism. In future releases of the presented approach, the data exchange between the sensors and actuators and the actual IoT simulation framework will be improved and further analyses results will be presented.

4.7 Conclusion

This work in progress paper provides the basis for an IoT simulation environment including HiL. The network simulator OMNeT++ is used to model the infrastructure between virtual and physical nodes. By extending its scheduler, extendibility with further HiL devices is enabled. Although it is too early to present further simulation results, the authors believe that the presented approach can help to encourage the research on IoT platforms to face the upcoming challenges, including the simulation and analysis of distributed and ubiquitous computing. As OMNeT++ was designed to model large networks, it is suitable for the simulation of future smart environments. The key to success is hereby the consideration of physical hardware in the virtual environment, as a simulator is not a universal solution and might fail in cases where real world data is required to gain feasible results.

The presented approach moreover features the functionality of a gateway, where the translation between different communication standards is not limited to a specific device. By using the intermediate representation, a decentralized mechanism is developed that transfers relevant data to the respective node where it is translated to the final communication standard.

In future, effort will be invested to rework the data flow from multiple sources to multiple destinations in complex networks. In example of a smart home environment, it will be demonstrated how the simulation of not available sensors and actuators can be handled by the presented framework. Also security aspects will be taken into account. Currently, no specific OMNeT++ library, like iNet, is used to model the network. In future versions of the presented approach this will become an essential part of the IoT simulator.

Acknowledgments This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 643892.

References

1. M. Weiser, The computer for the 21st century, https://www.ics.uci.edu/~corps/phaseii/Weiser-Computer21stCentury-SciAm.pdf
2. A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of things: a survey on enabling technologies, protocols and applications. IEEE Commun. Surv. Tutorials 17 (2015)

3. S. Schürmans, G. Onnebrink, R. Leupers, G. Ascheid, X. Chen, ESL power estimation using virtual platforms with black box processor models. Proceedings of the 3rd workshop on virtual prototyping of parallel and embedded systems (ViPES), Samos, Greece, July 2015
4. P. Ittershagen, P. Hartmann, K. Grüttner, W. Nebel, A workload extraction framework for software performance model generation. Proceedings of the 7th workshop on rapid simulation and performance evaluation: methods and tools (RAPIDO), Amsterdam, The Netherlands, Jan 2015
5. A. Varga, The OMNeT++ discrete event simulation system. Proceedings of the 2001 European simulation multiconference (ESM), Prague, Czech Republic, June 2001
6. Riverbed application and network performance management solutions, http://www.riverbed.com/products/performance-management-control/opnet.html
7. P. Parsch, A. Masrur, W. Hardt, Designing reliable home-automation networks based on unidirectional nodes. Proceedings of the 9th IEEE international symposium on industrial embedded systems (SIES), Pisa, Italy, June 2014
8. S. Sotiriadis, N. Bessis, E. Asimakopoulou, N. Mustafee, Towards simulating the internet of things. Proceedings of the 28th international conference on advanced information networking and applications workshops (WAINA), Victoria, BC, May 2014
9. E. Egea-López, F. Ponce-Marín, J. Vales-Alonso, A.S. Martínez-Sala, J. García-Haro, OBIWAN: wireless sensor networks with OMNET++. Proceedings of the 2006 IEEE Mediterranean electrotechnical conference (MELECON), Malaga, Spain, May 2006
10. Thread Group, http://threadgroup.org/
11. mediola – connected living AG, http://www.mediola.eu/
12. O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, I. Soler Jubert, M. Mazura, M. Harrison, M. Eisenhauer, P. Doody, Internet of things strategic research roadmap. Cluster of European Research Projects on the Internet of Things, http://www.internet-of-things-research.eu/pdf/IoT_Cluster_Strategic_Research_Agenda_2011.pdf
13. D. Michalek, C. Gehsat, R. Trapp, T. Bertram, Hardware-in-the-loop-simulation of a vehicle climate controller with a combined HVAC and passenger compartment model. Proceedings of the 2005 IEEE/ASME international conference on advanced intelligent mechatronics, Monterey, CA, July 2005
14. B. Aksun Güvenç, L. Güvenç, S. Karaman, Robust yaw stability controller design and hardware-in-the-loop testing for a road vehicle. IEEE Trans. Veh. Technol. **58**(2), 555–571 (2008)
15. P. Wehner, F. Schwiegelshohn, D. Göhringer, M. Hübner, Development of driver assistance systems using virtual hardware-in-the-loop. Proceedings of the 14th international symposium on integrated circuits (ISIC), Singapore, Dec 2014
16. EU project RADIO, Robots in assisted living environments: unobtrusive, efficient, reliable and modular solutions for independent ageing, http://www.radio-project.eu/
17. EnOcean GmbH, ECO 200 energy converter for motion energy harvesting, https://www.enocean.com/en/enocean_modules/eco-200/
18. EnOcean GmbH, EnOcean Pi transforms raspberry Pi into a wireless gateway, https://www.enocean.com/en/enocean-pi/
19. RaZberry, http://razberry.z-wave.me
20. FIBAR GROUP, FIBARO wall plug, http://www.fibaro.com/en/the-fibaro-system/wall-plug