Towards Comparable Simulations of Cooperating Objects and Wireless Sensor Networks

Thiemo Voigt, Joakim Eriksson, Fredrik Österlind
Swedish Institute of Computer Science
{thiemo,joakime, fros}@sics.se

Robert Sauter, Nils Aschenbruck, Pedro J. Marrón
University of Bonn
{sauter,aschenbruck, pjmarron}@cs.uni-bonn.de

Vinny Reynolds, Lei Shu
DERI, Galway, Ireland
{vinny.reynolds, lei.shu}@deri.org

Otto Visser
TU Delft
o.w.visser@tudelft.nl

Anis Koubaa
ISEP-IPP
aska@isep.ipp.pt

Andreas Köpke
TU Berlin
koepke@tkn.tu-berlin.de

ABSTRACT

Simulators are indispensable tools to support the development and testing of cooperating objects such as wireless sensor networks (WSN). However, it is often not possible to compare the results of different simulation tools. Thus, the goal of this paper is the specification of a generic simulation platform for cooperating objects. We propose a platform that consists of a set of simulators that together fulfill desired simulator properties. We show that to achieve comparable results the use of a common specification language for the software-under-test is not feasible. Instead, we argue that using common input formats for the simulated environment and common output formats for the results is useful. This again motivates that a simulation tool consisting of a set of existing simulators that are able to use common scenario-input and can produce common output which will bring us a step closer to the vision of achieving comparable simulation results.

1. INTRODUCTION

Cooperating Objects are, in the most general case, small computing devices equipped with wireless communication capabilities that are able to cooperate and organize themselves autonomously into networks of sensors, actuators and processing units to achieve a common task. One instance of cooperating objects are wireless sensor networks (WSN). The applications of cooperating objects are numerous including industrial automation, home control, transportation as well as healthcare and assisted living to name a few.

Simulators are indispensable tools to support the development and testing of cooperating objects. Simulations are commonly used for rapid prototyping which is otherwise very difficult due the restricted interaction possibilities with this type of embedded systems. Simulators are also used for the evaluation of new network protocols and algorithms and enable repeatability because they are independent of the physical world and its impact on the objects. Simulations also enable nonintrusive debugging at the desired level of detail.

As shown in this paper, there exists a large number of simulators for cooperating objects. Despite the large number of simulators, however, there is no single simulation environment that enables comparison of algorithms and applications on different sensor network platforms such as TinyOS and Contiki.

Therefore, the goal of our work is to constitute a common simulation platform that allows the community to share code and simulation results, thus making them comparable to one another. We discuss the requirements on such a common simulation platform and show that no current existing simulator fulfills all our requirements. Therefore, the platform we propose consists of a set of simulators that together fulfill desired simulator properties.

We stress that the use of a common simulation specification language, as an alternative to achieve comparable results, is not feasible in general, although it sounds attractive. One reason is that simulators are specific, and in some simulators, it is not possible to change parameters that are easy to change in other simulators. Here, we provide some typical examples. For instance, in MSPSim [17] – an instruction-level emulator for TelosB and related mote types – it is not possible to run the CPU at a speed of 100 MHz since MSP430 processors do not run at this speed. Another example for MSPSim is that it is not possible to simulate applications under the assumptions of more available memory since the memory of the MSP430 is constrained. Hence, a simulation of a larger memory requires a reprogramming of the emulator – including defining an alternative emulated microcontroller architecture – instead of changing an input parameter, which is possible for simulators operating at a higher abstraction layer. As another example, assume using a more generic MAC protocol simulator framework (e.g., MAC Simulator from Delft [26]) to compare the energy consumption of a MAC protocol simulated with this simulator with X-MAC which is not available in Delft's simulator. Thus, consider COOJA/MSPSim to simulate X-MAC. However, previous work has shown that a certain unicast optimization in X-MAC almost doubles performance [16]. In order to compare to the "right"

WSNPERF 2009, October 23, 2009 - Pisa, Italy. Copyright 2009 ICST 978-963-9799-70-7/00/0004 $5.00.

X-MAC version, a common simulation specification would have to enable the user to somehow identify these low-level implementation alternatives – an almost impossible task.

The discussion above implies, in particular the last point, that in general specifying simulations in a common specification language is not possible since the number of detail required is far too exhaustive.

Instead, we propose using common input formats for the simulated environment and common output formats for the results. This motivates that the simulation tool consists of a set of existing simulators that are able to use common scenario-input and can produce common output which will bring us a step closer to the vision of achieving comparable simulation results. We hope that the community is interested in participating in our effort to achieve this vision.

Our paper starts with an overview of existing simulations and evaluation tools in Section 2. Our goal is not to provide an exhaustive survey of all existing tools but to focus on WSN simulators and tools frequently used for the evaluation of cooperating objects. After defining important simulator properties we discuss possible information that can be described in common input and output formats in Section 4. In Section 5 we discuss our proposal for a simulation tool. We present the progress we have made towards this tool in the following sections before summarizing our conclusions in Section 9.

2. EXISTING SIMULATION AND EVALUATION TOOLS

In this section, we report on existing simulators for cooperating objects and wireless sensor networks. We have a slight focus on WSN simulators used within CONET but also present some simulators not used or developed within CONET. Furthermore, we present generic simulators that can be used for simulating cooperating objects, as well as emulators for cooperating objects. We do not include application-specific cooperating object simulators used for robotics such as Stage [2] and Gazebo [1] which are important part of the general field of cooperating objects. Finally, we briefly survey tools for modeling of cooperating object scenarios.

2.1 Generic Simulators

NS-2 [13] is a discrete event network simulator popular in academic research for simulation of routing and multicast protocols over wired and wireless networks. A wide range of network application, protocols and radio models is available from the core distribution and 3rd party extension. However, the support for wireless sensor network simulation is limited. *GloMoSim* [49] is a discreet event network simulator, written in parsec. It enables simulations of mobile ad hoc networks and wireless sensor networks and includes a variety of protocols and models for all the network layers.

OPNET Modeler [21] is a powerful discrete event network simulator that supports a large library of communication protocols mainly for computer networks. Although OPNET was not specifically designed for cooperating objects, recent works have contributed with new models for low-power wireless personal area networks (WPANs) taking advantage of OPNET's flexibility and modularity. In [24, 22], the authors proposed a simulation model of the IEEE 802.15.4 standard protocol, which is available as a subset of the open-ZB open-source toolset [3]. This model was mainly used to evaluate the performance of the beacon-enabled mode of the IEEE 802.15.4 standard. On the other hand, OPNET Modeler provides a simulation model of the IEEE 802.15.1/ZigBee stack starting from version 14.

OMNeT++ [46] is a discrete event simulation package written in C++, primarily developed for the simulation of computer networks and other distributed systems. The OMNeT++ simulation models are composed of hierarchically nested modules that intercommunicate with message passing. The core of OMNeT++ contains for example detailed IP, TCP, FDDI and Ethernet protocol models. A significant number of 3rd party modules is available which allow the use in WSN research by providing realistic wireless channel models, radio models, mobility models and various protocols.

2.2 Cooperating Objects Simulators

UbiWise [7] is the original pervasive computing simulator enabling the testing of pervasive services, implementation of protocols and integration of devices in a virtual 3-d environment. As a human-in-the-loop simulator, virtual devices can execute real application code, interact with external services, and accept human input providing a powerful tool for simulating pervasive computing environments. The *Lancaster* [31] simulator supports the integration of third party simulators to evaluate location-based applications. Network packets from applications under test are intercepted in a modified kernel and redirected through the NS-2 thus providing the network simulation functionality, whilst a web services interface allows third party simulators, such as mobility models to be integrated unto a unified simulation.

NetTopo [41] is an open source simulator and visualizer tool designed to test and validate algorithms for wireless sensor networks. This platform-independent tool provides a flexible architecture allowing node, topology and algorithm components to be easily replaced and an additional suite of graphical components allowing the visualisation and validation of experiments built using the NetTopo framework.

TOSSIM [28] is the simulator framework of TinyOS and the availability of the tailored compiler for this operating system is also used to enable the simulation support. This allows to use the same source code for simulation and deployment. TOSSIM's focus is on code written for the TinyOS operating system.

Castalia [9] is a generic WSN simulator not tied to a specific platform and hence it does not run deployable code. Instead it is designed to be highly tunable with realistic node behaviour for example relating to access of the radio. Castalia is based on OMNeT++.

2.3 Cooperating Objects Emulators

ATEMU [39] (Atmel EMUlator) is a cycle-by-cycle emulator for the the MICA2 platform except EEPROM and external data flash. Since the last release 0.4 is dated March 31st, 2004 ATEMU can be considered abandoned. *Avrora* [44] is written in Java and supports the Mica2 platform and has recently been extended to support the MicaZ platforms including the CC2420 radio chip [14]. Avrora provides also a wide range of monitors that automatically track, for example, function calls, power consumption, I/O registers, memory, radio packets or stack and print changes during the execution and/or generate a report after the program has completed execution.

2.4 COOJA and MSPSim

The COOJA simulator [36], a Java-based sensor network simulator was originally designed to simulate networks of nodes running

the Contiki operating system. COOJA has the ability to mix simulations of sensor devices at multiple abstraction levels. These levels are application level, OS level, and hardware level. In the application level the simulated nodes run the application logic reimplemented in Java – the native language of COOJA. In the OS level the nodes use the same code as real nodes, but compiled for the host machine running COOJA. Finally in the hardware level the nodes run the same compiled code that can be used in real nodes, e.g. the same system image. The hardware level is provided by MSPSim that emulates systems based on the MSP430 processor family. By using MSPSim underneath, COOJA allows simulated nodes to execute the same system image as the one used on the real nodes. The nodes at different abstraction levels communicate with each other using one of the three radio propagation models available in COOJA. COOJA uses an XML-based simulation configuration format, e.g., for specifying the topology.

MSPSim [17] is an instruction level emulator of MSP430-based sensor network nodes. MSPSim targets cycle accurate emulation of both the MSP430 CPU core and built-in peripherals such as timers, serial communication and analog to digital converters. Furthermore, MSPSim emulates external components such as the radio chip CC2420, sensors, and flash memories. MSPSim also provides emulation of complete sensor devices such as the Tmote Sky [38] and Scatterweb ESB [40].

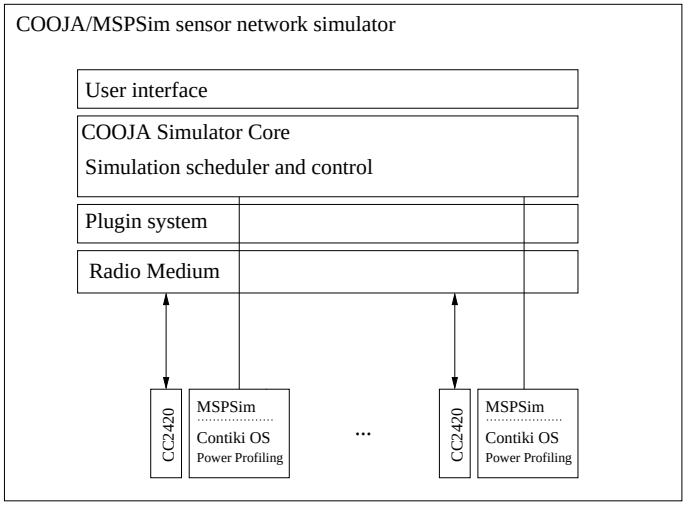

Figure 1: The architecture of the COOJA/MSPSim simulator. MSPSim is integrated into the COOJA simulator.

2.5 MiXiM

MiXiM is an integration effort that combines several OMNeT++ simulators. One of them is the Mobility Framework (MF) [15], the others are a MAC simulator [26], Positif (a simulator for localization) [27] and ChSim (a simulator targeting radio propagation models) [29].

Although different in goal, these four simulators were all running on OMNeT++ and had a lot in common so the idea was born to merge them all into one new simulator called MiXiM [23]. As the simulators had different foci, there was a limited amount of duplicate code to be dealt with and with the help of some glue code most of the code could be merged in a newly designed framework.

MiXiM simulations are specified using NED (network description language), the omnetpp.ini and an xml configuration file.

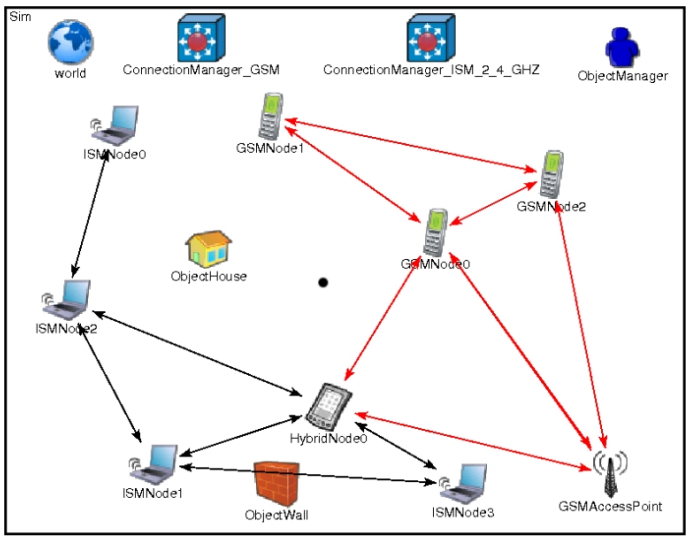

Figure 2: Illustration of a MiXiM Simulation

MiXiM is a modular framework, so the user can for example select a simple basic module for the MAC, choose a specific mobility model with waypoints and then focus testing on his/her new routing layer. Modules can be changed in the configuration without the need to recompile. There is a wide choice of modules available already, especially in mobility, localization and MACs.

The library of MAC protocol ranges from simple CSMA to IEEE 802.11. MiXiM inherits from the MAC Simulator wireless sensor network protocols including SMAC [48], TMAC [12] and Crankshaft [19]. There is also a wide choice on mobility models, including BonnMotion (see Section 2.6) support.

To ease the process of making new modules, all a user has to do is inherit from the appropriate base module and override a handful of methods. An overview of a MiXiM simulation, combining multiple radio frequencies and objects that influence radio propagation can be seen in Figure 2. For statistical analysis, modules can publish parameters in the utility module of a node. Other modules can then subscribe to the published data and transform this into readable and usable statistics. There is ongoing development in the area of placing objects/obstacles in the world and their effect on the radio propagation as well as battery models.

2.6 Tools for Scenario-Modeling

Simulation and emulation are techniques frequently used for performance evaluation of cooperating objects. If the objects are mobile, the movement patterns of these objects are found to have significant impact on the simulation and emulation results. This is quite obvious as the movements influence the topology of the network.

In the last decade, various synthetic models were proposed. There have been several general surveys [10, 8, 5, 33] as well as some specific ones for vehicular models [20]. Instead of providing details concerning the different models, a table of tools to generate synthetic mobility traces is provided (cf. Table 1). For all the tools listed it is possible to download a version on the respective website.

Tool		Models
Toilers-Code-Base	[10] [35, 34]	Random-Waypoint (sev. variants), Random-Walk, Prob. Random-Walk, Random-Direction, RPGM, Gauss-Markov, Column
BonnMotion	[45]	Random-Waypoint, Gauss-Markov, Manhattan-Grid, RPGM
Important	[6]	Random-Waypoint, RPGM, Freeway, Manhattan-Grid
MobiSim	[32]	Random-Waypoint, Random-Walk, RPGM, Gauss-Markov, Freeway, Manhattan
CanuMobiSim	[43] [42]	Brownian-Motion, Random-Waypoint, Meta-Model Obstacles
SUMO	[25]	Urban Vehicular Traffic

Table 1: Tools for generating synthetic mobility traces

In the rest of this section, we provide a brief overview on BonnMotion [45]. Within the simulation framework the tool BonnMotion was extended so that it could be integrated in different cooperating object simulators. Further details on this integrations will be described later.

BonnMotion is an open-source Java software which creates and analyzes mobility scenarios. It is developed at the University of Bonn, Germany, where it serves as a tool for the investigation of cooperating object scenario characteristics. The scenarios can also be exported for the network simulators ns-2, GloMoSim/QualNet, COOJA, and MiXiM.

Currently, there are five mobility models publicly available: Static, Random-Waypoint, Gauss-Markov, Manhattan-Grid, Reference Point Group Mobility (RPGM). Further scenarios such as the disaster area model [4] will be published shortly. For scenarios analysis different metrics can be calculated as *overall* statistics (averaged over the simulation time) and as *progressive* statistics (values of metrics for certain points in time). The following metrics are supported: relative mobility, average node degree, number of partitions, degree of separation, average link duration, average time to link break.

3. IMPORTANT PROPERTIES FOR SIMULATORS

We have identified the following requirements for cooperating objects simulators. Many requirements are derived from the CONET roadmap [30] while others are generally important simulator properties. We consider the following properties:

- *Heterogeneity*: it is important to enable the simulation of heterogeneous networks with respect to node platforms, simulated application codes and possibly operating systems,
- *Integration*: it is required to have an integrated support for high-fidelity radio propagation models, antenna models and physical-layer simulation. In addition, the integration of mobility models or the possibility to use traces generated by external mobility model tools is also important.
- *Scalability*: it is extremely important to support scalable simulations, due the large-scale nature of cooperating objects networks,
- *Extensibility*: it is mainly achieved by clearly defined APIs, scripting support and modular composition, and is a key design issue due to the fast evolving hardware and software used for cooperating objects. Additionally, a connection to robotics simulators requires a flexible basis to enable advanced integration.
- *Energy modeling*: Detailed estimation of power consumption and possibly simulation of battery models
- *Simulation of deployable code*: This is connected to the available support for the simulation of WSN platforms and operating systems, respectively. This is also an essential prerequisite for a tight integration between simulators and testbeds.
- *Ease of use*: it includes GUIs, introspection and debugging support and – although not core functionality of a simulator – is a very important element for the success with developers.

There is currently no simulator that is able to fulfill all these requirements. Therefore, a combination of existing approaches is necessary. TOSSIM is popular because of its tight integration with and the inclusion in the TinyOS distribution. However, the impossibility to simulate code for other operating systems limits TOSSIM's applicability. General network simulators such as ns-2 and projects built on top usually lack the possibility to simulate deployable code. From the available simulators, COOJA provides a very good basis as its core design supports the simulation of nodes at different abstraction levels ranging from emulation to abstract models implemented in JAVA. Additionally, within CONET, we have recently added support for using mobility traces. MiXiM provides more advanced support for mobility and for high-fidelity physical-layer simulation including detailed radio propagation models and antenna models.

Our experience of integrating different simulators into MiXiM and COOJA and MSPSim into one simulator has shown that the integration of simulators is a hard and tedious task. We therefore discuss in the next sections how we can in a useful way couple several simulation tools to make their results comparable.

4. COMMON INPUT/OUTPUT

In this section, we discuss several simulation environment parameters and simulation results that can be specified in a common format for several simulators. These formats can then either be translated to the formats of the individual simulators of the simulators can be adapted to the proposed formats. Using a common specification of these parameters is an important step towards archieving comparable results.

4.1 Common scenario-specific simulator input

As discussed above, a common simulation specification for a wide range of different simulators is not feasible. However, it would be useful to specify common scenario input where it is feasible. We identified the following issues as doable:

Topologies A description of the topology would help to make simulations more comparable in that nodes would be placed in

the same way in different simulation runs.

Node Movement In a cooperating objects scenario, nodes might move around, for example, robots might move or people might carry their mobile phones, sensors etc. with them. There are a number of mobility models for this task and there are also tools such as BonnMotion described in Section 2.6 that generate traces and log-files. As we will show later, both MiXiM and COOJA are able to read these log files and hence move nodes in simulations according to certain mobility models supported by BonnMotion. Node movement produced by the same tool based on the same mobility model reduces one uncertainty when comparing simulation results. One could see the description of node movement as an extension of topology description.

Sensor Readings There are a number of tools that can generate maps of physical phenomena, for example, fire simulators that predict how fire spreads in landscapes [11]. From these simulators a landscape with sensor inputs can be generated that can provide sensor nodes (or rather locations) with sensor input for any given instant. Combining this information with topology description of the node placement is useful.

Communication Models The communication/radio models are also useful. This might include transmission and interference ranges, but also more complex radio models or behaviour is desirable.

Radio Noise Radio noise is a special case of sensor reading and it can be treated in the same way as sensor readings.

Node failure Simulating node failure, which can result in holes forming in networks, dropped packets, incomplete algorithms, etc. is an important part of evaluating the robustness of protocols and algorithms implemented at various layers in cooperating objects. The accuracy of comparing simulation results generated using several tools is improved by ensuring that this can be modeled consistently within the simulation tools.

While at the first glance it sounds attractive to treat packet generation by the application for each node as something that could be specified, this is indeed problematic for some types of simulators even for simple applications. It might work for high-level simulators but is not always possible for simulators that simulate deployable code because it is seldom possible to generate the code for a given application behaviour. In more complex applications, the generation of packets from the application-level is a consequence of complex interactions that are not foreseeable before the simulation.

4.2 Common Output

There are a number of interesting output parameters that would make simulation results more comparable. These include but are not limited to the following:

Energy Consumption Energy is one of the primary concerns in cooperating objects and hence the energy consumption is of high interest. The output could be specified as energy consumption per node as average, minimum, maximum etc., as energy consumption per coverage area etc.

Packets The transmitted and received packets per node at the different layers are very interesting. It is also interesting to detail these for the different layers such as application, MAC and physical layer. The same is true for the received packets. Moreover, lost packets are of interest. This output may for example simplify comparisons between the energy-efficiency of different MAC protocols.

Channel utilization The usage of the channel over both time and space.

Based on these and other statistics interesting information can be inferred, e.g. preferred routing paths. An other important issue is that by having common output formats, the same tools can be used to analyze and process the simulation results.

In MiXiM, producing the common output format could be done using the publish/subscribe scheme in the utility module. The data is published by the protocol stack, and a specialized subscriber could be used to convert it into the common output format. We will evaluate this possibility in future work.

4.3 Common Input and Output

While common input and common output are useful, it is the combination of common input and output that can help to make simulation results comparable. If only the input is common, it is cumbersome and maybe even invalid to compare the results. If only the output is common, then there can be a lot of redundant work required to set up the simulations in a way that make the results comparable. If, however, both input and output are common, we expect that in many cases we will be able to directly compare results achieved with different simulators. It will also be possible to define common input scenarios and compare the results. We expect that in many cases the results of executions in different simulators will differ far more than expected. In such cases, the common output might help to understand why the results differ. For example, we might discover that the number of messages sent by the applications running on the nodes in a certain area might differ, although the sensor readings in the area are the same. In this example, common input will ensure that the sensors receive the same inputs whilst the common outputs will provide users with a means to discover and potentially debug any problems. For example, that a node has a reduced power level.

5. SPECIFICATION OF THE SIMULATION PLATFORM

Based on the discussion above, we propose a simulation tool that consists of common scenario-specific input and common output formats for existing cooperating object simulators.

Figure 3 shows the simulation platform we propose. Note that the simulation platform is generic enough to include most of the existing simulators. We hope that as many simulator developers as possible will join us to define useful formats and make their simulators part of the platform. Nevertheless, we have highlighted COOJA/MSPSim and MiXiM since these simulators together are able to fulfill the important properties outlined in Section 3. In particular, they are useful for the simulation of non-functional properties identified in the CONET Roadmap [30]. Furthermore, MiXiM and COOJA/MSPSim are both simulators that themselves integrate a number of simulators: MiXiM combines several simulation framework into one, namely the mobility framework [15], radio prop-

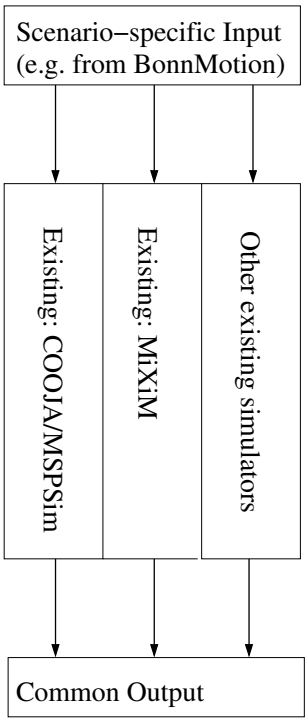

Figure 3: Specification of the Simulation Platform

agation models from the CHannel SIMulator (ChSim) [29], the MAC simulator [26], and the Positif framework [27]. COOJA integrates COOJA, MSPSim, Avrora [44] and has been made compliant with BonnMotion output. The latter is also true for MiXiM. COOJA/MSPSim's ability to run Contiki and TinyOS nodes in the same simulation runs enables also the comparison of the performance of the same application developed for these different operating systems [18].

5.1 Fulfillment of requirements and Steps Towards Realization

In this section we provide a table that shows that the specified simulation tool with MiXiM and COOJA/MSPSim fulfills the requirements and give an overview on the progress of the implementation that will be described later in more details.

5.1.1 Fulfillment of Requirements

Requirement	Simulator fulfilling
Heterogeneous networks	COOJA/MSPSim
High fidelity radio models antenna models physical layer simulation	MiXiM
Mobility models	MiXiM and COOJA/MSPSim via BonnMotion
Accurate power consumption	COOJA/MPSim
Scalability	MiXiM
Simulation of deployable code	COOJA/MSPSim
Extensibility	MiXiM and COOJA/MSPSim

Table 2: The specified tool fulfills the requirements

Table 2 shows that the specified simulation tool with MiXiM and COOJA/MSPSim fulfills the requirements that we have outlined in Section 3. As stated above, COOJA/MSPSim is able to execute simulations of Contiki and TinyOS nodes in the same simulation runs [18]. MiXiM's advanced capabilities of physical-layer simulations are discussed in the next section. Section 7 and 8 present the integration of BonnMotion output into COOJA and MiXiM. Another important related feature is that using COOJA/MSPSim and Contiki we can perform sensor network checkpointing [37]. In this approach every node exists both in testbed and simulation so that we can checkpoint and transfer network state between the two domains. This approach benefits from advantages of both simulation and testbeds: nonintrusive execution details, repeatability, and realism.

Again, we want to stress that we hope that also other simulator developers are interested in defining formats and making their simulators compatible.

6. MIXIM: ADVANCED PHYSICAL-LAYER SIMULATIONS

Simulating the physical layer of wireless communication remains a challenge. Communication standards like 802.11b and Bluetooth systems go beyond the simple single narrow frequency band, single antenna model used in popular simulators. Yet, these technologies gain popularity, since they provide researchers with a plethora of possibilities that can be explored to invent new protocols or improve existing ones. However, building a detailed and sufficiently accurate model for such complex systems is a tremendous task that takes a lot of time.

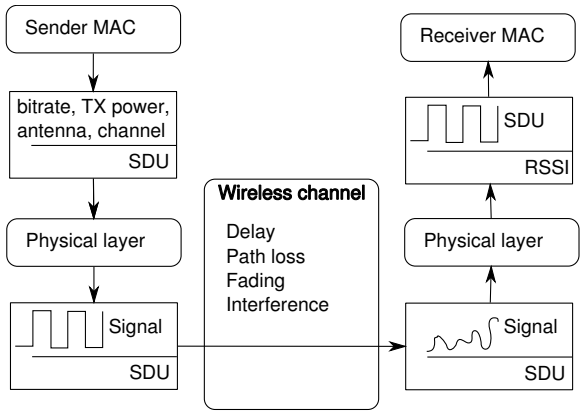

Figure 4: Aspects of a wireless transmission

In order to gain a deeper understanding of the complexity of the problem, let us start with the main components that are responsible for the transmission process, shown in Figure 4. In this figure, we consider the transmission of a single packet, concentrating on the interaction of the components. Assume, the sending Medium Access Control (MAC) protocol as handed a packet that shall be transmitted. After packing this into a Service Data Unit (SDU), the MAC protocol hands this packet down to the physical layer for transmission, together with some information on how the packet shall be transmitted. The physical layer uses this information to compute a signal that represents the packet. This signal is transmitted via the wireless medium, where it gets distorted by multiple influences, starting from attenuation due to various causes as well as other interfering transmitters. The physical layer of the receiver receives this distorted signal and has to derive a binary represen-

tation that hopefully resembles the original SDU. This is passed to the receiving MAC layer, together with some meta information like the Received Signal Strength Indicator (RSSI) of the packet.

While many popular simulators only model single frequency, single antenna systems, such models are not sufficient anymore. The physical layer of MiXiM is designed with flexibility in mind without sacrificing efficiency. It can be used for simple single frequency, single bit rate systems used for instance in sensor network simulations, as well as for multiple channels, multiple bitrate systems like IEEE 802.11b that sends the header and the payload of the packet with different bit rates. It can be used for systems that change the transmission frequency between each packet like Bluetooth, Orthogonal Frequency Division Multiplexing (OFDM) systems like 802.11a that transmit in parallel on multiple frequencies and Multiple Input Multiple Output (MIMO) systems that use multiple antennas for the transmission and the reception.

In addition to the wide range of different transmission standards, there are many different models for the wireless channel, each concentrating on a different effect in a certain environment. There are path loss models that attenuate the transmitted signals according to the traveled distance, abstract models for shadowing effects due to obstacles like the log-normal fading, models for fast fading due to the mobility of the nodes like Rice and Rayleigh fading and many more.

To complicate things further, standards like 802.11g include Forward Error Correction (FEC) and the design of the physical layer of MiXiM should not prevent research on the influence of such codes. This means that all the attenuation effects should be computable on a sub-packet time scale.

To address these issues, the physical layer of MiXiM was redesigned to support the modeling of complex signals in both time, frequency and space. Further details can be found in [47].

7. INTEGRATION OF BONNMOTION WITH COOJA AND MIXIM

The native format in which BonnMotion saves the movement traces is node-by-line waypoint based. This means that there is one line for each node. This line contains all the waypoints. A waypoint is a position at which the movement of a node (e.g. direction, velocity) changes. A waypoint consists of:

- the simulation time in seconds at which the waypoint is reached by the node
- the x and y coordinates of the position of the waypoint

This format implies that during the simulations for each event the current node positions have to be calculated based on the waypoints. If there are many events, this may have a negative impact on the runtime of a simulation. An alternative is to use an interval based approach. The nodes are regarded as stationary for an interval. The positions of the nodes are updated periodically after each interval by a specific position update event. By doing so, the current node positions do not have to be calculated for each event. However, the number events is increased, which may also influence the runtime of a simulation negatively. A factor that has a major impact in this context is the interval length. Smaller intervals yield higher accuracy but also more events. Overall, it is a trade-off between the number of events and the runtime per event.

Within CONET, BonnMotion was extended to support the interval based trace format. Trace files in the BonnMotion's native trace format can be transformed to an interval-based format using the *IntervalFormat* option. The interval length can be specified using the *-I* option. The default value is one second. The interval trace format is an interval-by-line based. This means that there is one line for each interval of each node. A line consists of:

- the node number
- the simulation time in seconds (in intervals)
- the x and y coordinates of the position of the node for the interval

The interval based trace format is used by COOJA and MiXiM.

8. EVALUATION OF THE INTEGRATION OF BONNMOTION AND COOJA

The COOJA/BonnMotion integration is implemented as an optional COOJA plugin. The full code will be available for public use in the next Contiki/COOJA release.

The COOJA plugin accepts BonnMotion output files, which contains mote-time specific locations. A BonnMotion location file is parsed by COOJA at simulation setup, and the resulting node moves are scheduled in the COOJA simulation loop.

The execution time overhead of adding mobility to COOJA majorly depends on two factors: how often nodes move, and the graphical COOJA setup. When running COOJA in graphical mode, the node positions are reflected in visualizers allowing a user to see how and where nodes move. In contrast, when COOJA is running in non-graphical batch-mode, the execution time overhead of moving nodes is reduced due to the lack of active visualizers.

We evaluate the mobility execution time overhead with respect to both the simulation size and the interval at which nodes have moved. We simulate Contiki networks at the operating system level, in contrast with emulating nodes using MSPSim. Since emulated nodes are more computationally demanding, emulated networks will have a lower relative mobility overhead than in our test setup. All tests are run in COOJA's batch-mode.

The simulated Contiki application is simple: all nodes broadcast a radio packet every two seconds. We simulate network sizes between 100 to 1000 nodes, and mobility intervals on the order of milliseconds. At each mobility interval, all simulated nodes are moved according to the specified BonnMotion file.

The execution overhead of mobility is low, even when the nodes are moved once every second. Figure 5 shows the execution times with and without mobility for a 100-node network simulated for one minute. The mobility interval is one second: all nodes move every second. As shown in the figure, the execution duration of simulating 600 nodes for 60 simulated seconds is around 60 seconds, and the mobility overhead is negligible. The exponential increase in execution time is due to COOJA's current event queue implementation: the linked list implementation is not optimized for large queue sizes.

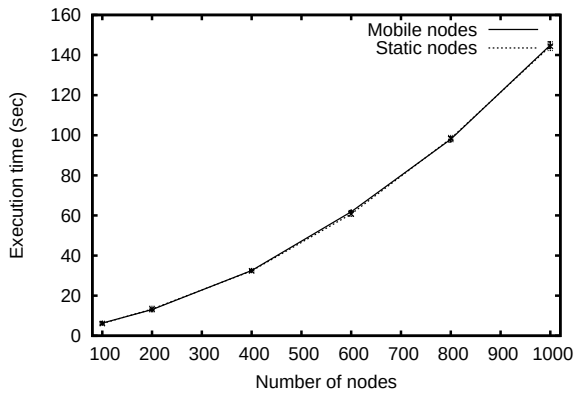

Figure 5: Impact on number of mobile nodes on performance

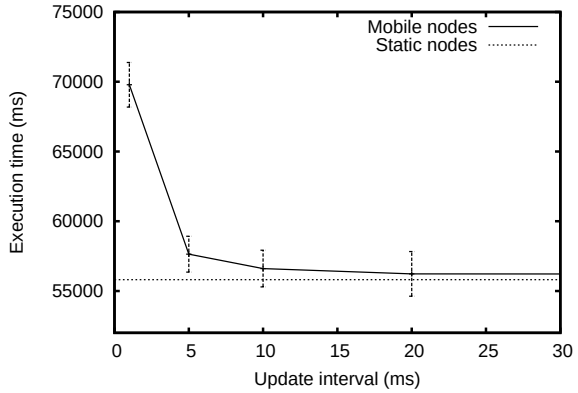

Figure 6: Impact on mobility rate update on performance

To further study the execution overhead, we gradually increase the mobility update rate. Each simulated network consists of 100 nodes and runs for 10 simulated minutes. The execution time without mobility is 55.8 seconds. As seen in Figure 6, the execution time increases as the mobility update interval decreases. The overhead is still low (1.43%) with a 10 ms update interval, but greatly increases at higher update rates. It should be noted that such high update rates - hundreds of moves per second - are typically not necessary in simulations of cooperating objects.

Comparable results could be shown for MiXiM, but since this does not add additional insight, they are not shown here. The main limits for simulation speed is IO load, since the file has to be transferred from the disk into memory and parsing overhead.

9. CONCLUSION AND FUTURE WORK

In this paper, we have specified a simulation platform that aims at fulfilling the vision of enabling comparable simulation results for cooperating objects' simulators. We have shown that the use of a common specification language to describe the scenario configuration, the input parameters and the output statistics is a very promising alternative to achieve comparable results on different simulation platforms. Our next goal is to show that we can actually achieve more comparable simulation results. In addition, we expect that the community is interested in participating in our effort.

Acknowledgments

This work has been partially supported by CONET, the Cooperating Objects Network of Excellence, funded by the European Commission under FP7 with contract number FP7-2007-2-224053.

10. REFERENCES

[1] http://playerstage.sourceforge.net/doc/gazebo-manual-0.8.0-pre1-html/.

[2] http://playerstage.sourceforge.net/doc/stage-3.0.1/.

[3] Open-source ieee 802.15.4 opnet simulation model.

[4] N. Aschenbruck, E. Gerhards-Padilla, M. Gerharz, M. Frank, and P. Martini. Modelling Mobility in Disaster Area Scenarios. In *Proc. of the 10th ACM-IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 4–12, 2007.

[5] F. Bai and A. Helmy. A survey of mobility models, 2004. `http://nile.usc.edu/~helmy/important/Modified-Chapter1-5-30-04.pdf`.

[6] F. Bai, N. Sadagopan, and A. Helmy. IMPORTANT: A framework to systematically analyze the Impact of Mobility on Performance of RouTing protocols for Adhoc NeTworks. In *Proc. of the IEEE Infocom*, pages 825–835, 2003.

[7] J. Barton and V. Vijayaraghavan. UBIWISE, A simulator for ubiquitous computing systems design. *Hewlett-Packard Laboratories Palo Alto,âĂİ HPL-2003-93*, 2003.

[8] C. Bettstetter. Mobility modeling in wireless networks: categorization, smooth movement, and border effects. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(3):55–66, 2001.

[9] A. Boulis. Castalia: revealing pitfalls in designing distributed algorithms in WSN. In *Proceedings of the 5th international conference on Embedded networked sensor systems (Demo session)*, Nov. 2007.

[10] T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communication and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, Sep. 2002.

[11] J. Coleman and A. Sullivan. A real-time computer application for the prediction of fire spread across the Australian landscape. *Simulation*, 67(4):230, 1996.

[12] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *1st ACM Conf. on Embedded Networked Sensor Systems (SenSys 2003)*, pages 171–180, Los Angeles, CA, USA, Nov. 2003.

[13] DARPA. The network simulator, ns-2.

[14] R. de Paz Alberola and D. Pesch. AvroraZ: extending Avrora with an IEEE 802.15. 4 compliant radio chip model. In *Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, 2008.

[15] W. Drytkiewicz, S. Sroka, V. Handziski, A. Koepke, and H. Karl. A mobility framework for OMNeT++. In *3rd International OMNeT++ Workshop*, 2003.

[16] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the Fourth Workshop on Embedded Networked Sensors (Emnets IV)*, Cork, Ireland, June 2007.

[17] J. Eriksson, A. Dunkels, N. Finne, F. Österlind, F. T. Voigt, and N. Tsiftes. Demo abstract: MSPsim - an extensible simulator for MSP430-equipped sensor boards. In *European*

Conference on Wireless Sensor Networks (EWSN 2008), Demo Abstract, Bologna, Italy, Jan. 2008.

[18] 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 *Proceedings 2nd International Conference on Simulation Tools and Techniques (SIMUTOOLS'09)*, Rome, Italy, Mar. 2009.

[19] G. Halkes and K. Langendoen. Crankshaft: An energy-efficient MAC-protocol for dense wireless sensor networks. In *4th European conference on Wireless Sensor Networks (EWSN'07)*, pages 228–244, Delft, The Netherlands, Jan. 2007.

[20] S. P. Hoogendoorn and P. H. L. Bovy. State-of-the-art of vehicular traffic flow modelling. *Journal of Systems and Control Engineering - Special Issue on Road Traffic Modelling and Control*, 215(4):283–304, 2001.

[21] O. T. Inc. Opnet Modeler - ver. 11.5a.

[22] P. Jurcik, A. Koubaa, M. Alves, E. Tovar, and Z. Hanzalek. Simulation model for the IEEE 802.15.4 protocol: Delay/throughput evaluation of the GTS mechanismİ. In *15th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTSÂt'07), Istanbul, Turkey*, Oct. 2007.

[23] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. Klein Haneveld, T. Parker, O. Visser, H. Lichte, and S. Valentin. Simulating Wireless and Mobile Networks in OMNeT++: The MiXiM Vision. In *First International OMNeT++ Developers Workshop, Marseille, France*, Mar. 2008.

[24] A. Koubaa, M. Alves, and E. Tovar. A comprehensive simulation study of slotted CSMA/ca for IEEE 802.15.4 wireless sensor networks. In *6th IEEE Workshop on Factory Communication Systems (WFCS'06), Torino (Italy)*, pages 183–192, June 2006.

[25] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner. Sumo (simulation of urban mobility); an open-source traffic simulation. In *Proceedings of the 4th Middle East Symposium on Simulation and Modelling (MESM2002)*, pages 183–187, 2002.

[26] K. Langendoen et al. Mac simulator.

[27] K. Langendoen et al. Positif localization simulation framework.

[28] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 126–137, 2003.

[29] H. Lichte and S. Valentin. Implementing MAC protocols for cooperative relaying: A compiler-assisted approach. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems (SIMUTools)*, 2008.

[30] Marron et al. Research roadmap on cooperating objects, 2009. in print.

[31] R. Morla and N. Davies. Evaluating a location-based application: A hybrid test and simulation environment. In *Proceedings of 2nd International Conference on Pervasive Computing*, 2004.

[32] S. M. Mousavi, H. R. Rabiee, M. Moshref, and A. Dabirmoghaddam. Mobisim: A framework for simulation of mobility models in mobile ad-hoc networks. In *Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, 2007.

[33] M. Musolesi and C. Mascolo. Mobility models for systems evaluation. *State of the Art on Middleware for Network Eccentric and Mobile Applications (MINEMA)*, 2008.

[34] W. Navidi and T. Camp. Stationary distributions for the random waypoint mobility model. *IEEE Transactions on Mobile Computing*, 3(1):99–108, Jan-Feb 2004.

[35] W. Navidi, T. Camp, and N. Bauer. Improving the Accuracy of Random Waypoint Simulations through Steady-State Initialization. In *Proceedings of the 15th International Conference on Modeling and Simulation*, pages 319–326, 2004.

[36] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006), Tampa, Florida, USA*, Nov. 2006.

[37] F. Österlind, A. Dunkels, T. Voigt, N. Tsiftes, J. Eriksson, and N. Finne. Sensornet checkpointing: Enabling repeatability in testbeds and realism in simulators. In *Proceedings of the 6th European Conference on Wireless Sensor Networks, EWSN 2009*, Cork, Ireland, Feb. 2009.

[38] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proc. IPSN/SPOTS'05*, Los Angeles, CA, USA, Apr. 2005.

[39] J. Polley, D. Blazakis, J. McGee, D. Rusk, and J. Baras. Atemu: A fine-grained sensor network simulator. In *Proceedings of the First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, 2004.

[40] J. Schiller, H. Ritter, A. Liers, and T. Voigt. Scatterweb - low power nodes and energy aware routing. In *Proceedings of Hawaii International Conference on System Sciences*, Hawaii, USA, 2005.

[41] L. Shu, C. Wu, Y. Zhang, J. Chen, L. Wang, and M. Hauswirth. Nettopo: Beyond simulator and visualizer for wireless sensor networks. In *The Second International Conference on Future Generation Communication and Networking (FGCN 2008), Hainan, China, December 13-15*, 2008.

[42] A.-K. Souley and S. Cherkaoui. Advanced mobility models for ad hoc network simulations. In *Proceedings Systems Communications*, pages 50–55, 2005.

[43] I. Stepanov, J. Hähner, C. Becker, J. Tian, and K. Rothermel. A Meta-Model and Framework for User Mobility in Mobile Networks. In *Proc. of the 11th Int. Conf. on Networking 2003 (ICON 2003)*, pages 231–238, 2003.

[44] B. Titzer, D. Lee, and J. Palsberg. Avrora: scalable sensor network simulation with precise timing. In *Proceedings of the 4th international symposium on Information processing in sensor networks (IPSN)*, Apr. 2005.

[45] University of Bonn. *BonnMotion - a mobility scenario generation and analysis tool*, 2005. `http://bonnmotion.iv.cs.uni-bonn.de/`.

[46] A. Varga. Using the OMNeT++ discrete event simulation system in education. *IEEE Transactions on Education*, 42:372, 1999.

[47] K. Wessel, M. Swigulski, A. Köpke, and D. Willkomm. MiXiM - the physical layer: An architecture overview. In *Proceeding of the 2. International Workshop on OMNeT++*, Rome, Italy, Mar. 2009.

[48] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated, adaptive sleeping for wireless sensor

networks. *IEEE/ACM Trans. on Networking*, 12(3):493–506, 2004.

[49] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: a library for parallel simulation of large-scale wireless networks. *ACM SIGSIM Simulation Digest*, 28(1):154–161, 1998.