COOJA/MSPSim: Interoperability Testing for Wireless Sensor Networks

Joakim Eriksson, Fredrik Österlind, Niclas Finne, Nicolas Tsiftes, Adam Dunkels, Thiemo Voigt
Swedish Institute of Computer Science
{joakime,fros,nfi,nvt,adam,thiemo}@sics.se

Robert Sauter, Pedro José Marrón
University of Bonn and Fraunhofer IAIS
{sauter,pjmarron}@cs.uni-bonn.de

ABSTRACT

Wireless sensor networks are moving towards emerging standards such as IP, ZigBee and WirelessHART which makes interoperability testing important. Interoperability testing is performed today through black-box testing with vendors physically meeting to test their equipment. Black-box testing can test interoperability but gives no detailed information of the internals in the nodes during the testing. Black-box testing is required because existing simulators cannot simultaneously simulate sensor nodes with different firmware. For standards such as IP and WirelessHART, a white-box interoperability testing approach is desired, since it gives details on both performance and clues about why tests succeeded or failed. To allow white-box testing, we propose a simulation-based approach to interoperability testing, where the firmware from different vendors is run in the same simulator.

We extend our MSPSim emulator and COOJA wireless sensor network simulator to support interoperable simulation of sensor nodes with firmware from different vendors. To demonstrate both cross-vendor interoperability and the benefits of white-box interoperability testing, we run the state-of-the-art Contiki and TinyOS operating systems in a single simulation. Because of the white-box testing, we can do performance measurement and power profiling over both operating systems.

Categories and Subject Descriptors

I.6.4 [**Computing Methodologies**]: Simulation and Modeling Model Validation and Analysis; D.2.5 [**Software**]: Software Engineering Testing and Debugging

Keywords

Wireless Sensor Networks, Simulation, Interoperability

General Terms

Experimentation, Standardization, Measurement

1. INTRODUCTION

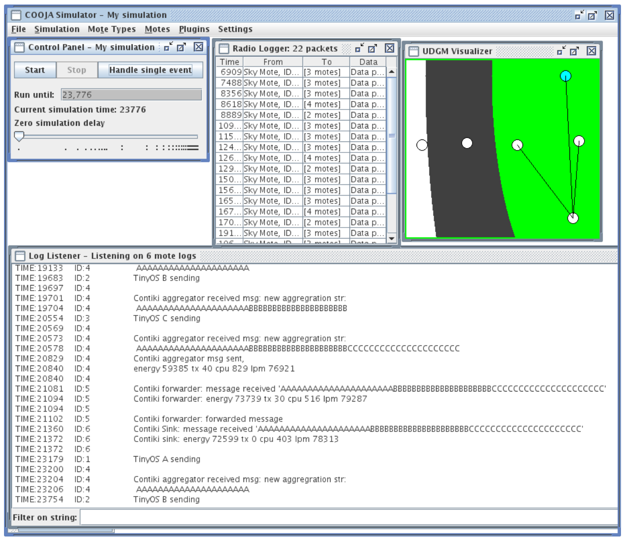
Wireless sensor networks are distributed systems consisting of small, often battery-powered sensing devices that communicate using low-power wireless radios. Wireless sensor networks enable numerous applications ranging from water [25] and bridge monitoring [19] to predictive maintenance in industry [16].

A vast body of research on sensor networks has yielded numerous different communication architectures and protocols. The rapid acceptance of sensor networks for industrial applications has, however, driven focus towards standardization and interoperability. Due to the proven industrial potential of wireless sensor networks, a number of standards have emerged during recent years. These include WirelessHART [15], ZigBee [26], and IPv6-based sensor networks [9].

The standardization of sensor networks makes interoperability a new requirement when implementing communication stacks and applications. Interoperability is difficult to verify, however, since sensor network simulators and prototyping tools lack the possibility to simulate heterogeneous sensor networks. For example, the widely used TOSSIM simulator [18] only simulates nodes running the TinyOS operating system [14]. These tools are thus unable to test the requirements that have arisen with the vision of an Internet of Things [8]. Instead, vendors today are required to physically meet to perform black-box testing of their equipment.

To meet the need for white-box testing of interoperability, we extend the Contiki simulator COOJA [21] and MSPSim, an instruction level emulator that is integrated into COOJA. Our simulator extensions enable simulations of heterogeneous sensor networks that consist of different operating systems and sensor devices. We also extend MSPSim with a power profiling mechanism similar to the software-based power profiler of the Contiki operating system [6]. Our experiments demonstrate that our combined simulator is not only a feasible tool for interoperability testing; it is also able to power profile nodes running different operating systems.

We evaluate our simulator with two state-of-the-art sensor node operating systems: Contiki and TinyOS. The Contiki operating system was the first operating system to support


COOJA Simulator – My simulation
File Simulation Mote Types Motes Plugins Settings
Control Panel – My simulation
Start
Stop
Handle single event
Run until: 23,776
Current simulation time: 23776
Zero simulation delay
Radio Logger: 22 packets
UDGM Visualizer
Log Listener – Listening on 6 mote logs
Filter on string:

Time	From	To	Data
6909	Sky Mote, ID...	[3 motes]	Data p...
7488	Sky Mote, ID...	[3 motes]	Data p...
8356	Sky Mote, ID...	[3 motes]	Data p...
8618	Sky Mote, ID...	[4 motes]	Data p...
8889	Sky Mote, ID...	[2 motes]	Data p...
109...	Sky Mote, ID...	[3 motes]	Data p...
115...	Sky Mote, ID...	[3 motes]	Data p...
124...	Sky Mote, ID...	[3 motes]	Data p...
126...	Sky Mote, ID...	[4 motes]	Data p...
129...	Sky Mote, ID...	[2 motes]	Data p...
150...	Sky Mote, ID...	[3 motes]	Data p...
156...	Sky Mote, ID...	[3 motes]	Data p...
165...	Sky Mote, ID...	[3 motes]	Data p...
167...	Sky Mote, ID...	[4 motes]	Data p...
170...	Sky Mote, ID...	[2 motes]	Data p...
191...	Sky Mote, ID...	[3 motes]	Data p...

```
TIME:19133   ID:4     AAAAAAAAAAAAAAAAAAAA
TIME:19683   ID:2     TinyOS B sending
TIME:19697   ID:4
TIME:19701   ID:4     Contiki aggregator received msg: new aggregration str:
TIME:19704   ID:4     AAAAAAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBBBBB
TIME:20554   ID:3     TinyOS C sending
TIME:20569   ID:4
TIME:20573   ID:4     Contiki aggregator received msg: new aggregration str:
TIME:20578   ID:4     AAAAAAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBBBBBCCCCCCCCCCCCCCCCCCCC
TIME:20829   ID:4     Contiki aggregator msg sent,
TIME:20840   ID:4     energy 59385 tx 40 cpu 829 lpm 76921
TIME:20840   ID:4
TIME:21081   ID:5     Contiki forwarder: message received 'AAAAAAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBBBBBCCCCCCCCCCCCCCCCCCCC'
TIME:21094   ID:5     Contiki forwarder: energy 73739 tx 30 cpu 516 lpm 79287
TIME:21094   ID:5
TIME:21102   ID:5     Contiki forwarder: forwarded message
TIME:21360   ID:6     Contiki Sink: message received 'AAAAAAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBBBBBCCCCCCCCCCCCCCCCCCCC'
TIME:21372   ID:6     Contiki sink: energy 72599 tx 0 cpu 403 lpm 78313
TIME:21372   ID:6
TIME:23179   ID:1     TinyOS A sending
TIME:23200   ID:4
TIME:23204   ID:4     Contiki aggregator received msg: new aggregration str:
TIME:23206   ID:4     AAAAAAAAAAAAAAAAAAAA
TIME:23754   ID:2     TinyOS B sending
```

Figure 1: COOJA simulating a hierarchical, heterogeneous sensor network of TinyOS and Contiki nodes

IP for sensor networks [3] and is currently the only system that fulfills all the IPv6 Ready compliance requirements [9]. The TinyOS operating system is popular in academia and has been used to implement standard protocols [13].

The contribution of this paper is the design and evaluation of a heterogeneous simulator environment for interoperability testing, with which we demonstrate how white-box testing enables accurate and non-intrusive power profiling of different operating systems.

The rest of this paper is structured as follows. We describe the two simulators that we combine and the operating systems that we simulate in our experiments in Section 2. We outline the technical aspects of our implementation in Section 3. In Section 4, we evaluate the accuracy and ability of COOJA/MSPSim in heterogeneous sensor network simulations. Thereafter we discuss related work in Section 5, and conclude the paper in Section 6.

2. BACKGROUND

Our white-box interoperability testing system builds on the COOJA [21] and MSPSim simulators [10]. COOJA is a cross-level sensor network simulator. MSPSim can be used through COOJA to emulate sensor devices based on the popular MSP430 processor. We describe each of these two simulators as well as the TinyOS and Contiki operating systems that we use as examples for interoperability testing.

2.1 The MSPSim Simulator

MSPSim [10] is a Java-based instruction level emulator of the MSP430 microprocessor series. In contrast with CPU-level emulators, it emulates complete sensor networking platforms such as the Tmote Sky [22] and ESB/2 [23]. MSPSim targets both realistic simulation with accurate timing for use as a research tool, and good debugging support for use as a development tool.

MSPSim combines cycle accurate interpretation of CPU instructions with a discrete-event based simulation of all other components, both internal and external. MSPSim uses an event-based execution kernel that enables accurate timing while keeping the host processor utilization as low as possible. We show an outline of the main execution loop in Figure 2. Before interpreting instructions, MSPSim executes all pending events in both event queues. Each queue handles events that are scheduled with a different perspective of time, with the first being based on CPU clock cycles, whereas the other is based on a high resolution clock. Most of the internal components of the MSP430, such as the USART and the analog-to-digital converter use the event queue for clock cycles, while external components such as radio transceivers use the event queue for the the high resolution clock.

```
while (running) {
  /* execute events */
  executeCycleEvents(cycles);
  executeTimeEvents(currentTime);

  /* fetch instruction to execute */
  op = memory[pc++] | (memory[pc++] << 8)
  instruction = decodeInstruction(op);
  switch(instruction) {
    case MOV:
      dst = readSrc(op);
      cycles += MOV_CYCLES;
      break;
    case ADD:
      ...
  }
}
```

Figure 2: The core emulation loop of MSPSim in pseudo-code. After handling queued events, the program counter (PC) is increased and the next instruction is decoded and executed.

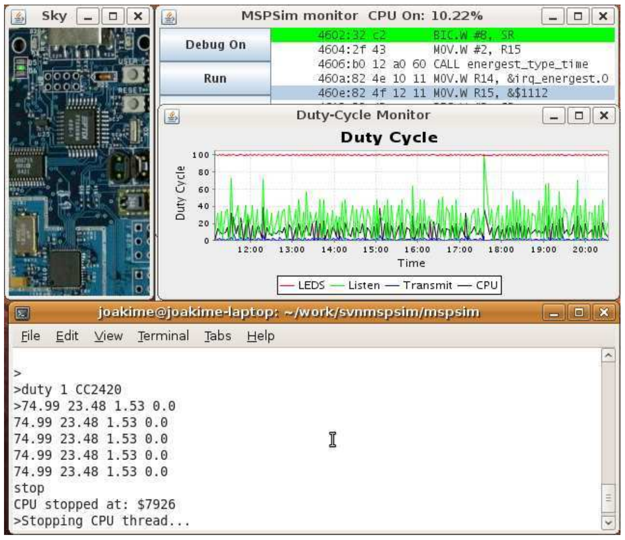
The emulator provides a programming interface for integration with simulation frameworks such as COOJA. In addition, the emulator can be extended with new mote types through a mote interface and I/O interfaces that correspond to the MSP430 I/O ports and serial communication ports.

MSPSim provides both debugging capabilities such as break points, watches, logging, and single stepping as well as statistics about the operating modes of the emulated components, statistics such as how much time the CPU has consumed in the different low-power modes. Figure 3 shows the MSPSim interface for viewing component duty cycles. All features and information can be accessed either via a command line interface, or via the integration programming interfaces.

2.2 The COOJA Simulator

COOJA [21] is a flexible Java-based simulator initially designed for simulating networks of sensors running the Contiki operating system. COOJA simulates networks of sensor nodes where each node can be of a different type; differing not only in on-board software, but also in the simulated hardware. COOJA is flexible in that many parts of the simulator can be easily replaced or extended with additional functionality.

A simulated node in COOJA has three basic properties: its data memory, the node type, and its hardware periph-



```
4602:32 c2       BIC.W #8, SR
4604:2f 43       MOV.W #2, R15
4606:b0 12 a0 60 CALL energest_type_time
460a:82 4e 10 11 MOV.W R14, &irq_energest.0
460e:82 4f 12 11 MOV.W R15, &$1112
```

```
>
>duty 1 CC2420
>74.99 23.48 1.53 0.0
74.99 23.48 1.53 0.0
74.99 23.48 1.53 0.0
74.99 23.48 1.53 0.0
74.99 23.48 1.53 0.0
stop
CPU stopped at: $7926
>Stopping CPU thread...
```

Figure 3: MSPSim emulating a Tmote Sky node running a program that periodically turns on the radio and sends data packets.

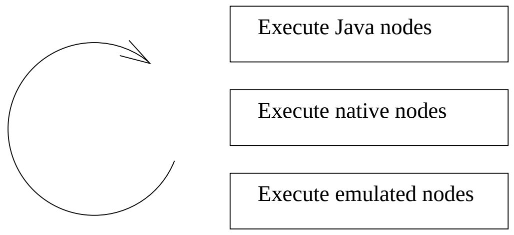


Figure 4: The main loop in COOJA executes nodes at different levels.

erals. The node type may be shared between several nodes and determines properties common to all these nodes. For example, nodes of the same type run the same program code on the same simulated hardware peripherals. Nodes of the same type are initialized with the same data memory, except for the node id. During execution, however, the data memories of the nodes will eventually differ after reacting to external stimuli.

COOJA can execute Contiki programs in two different ways. Either by running the program code as compiled native code directly on the host CPU, or by running compiled program code in MSPSim. COOJA is also able to simulate nodes developed in Java at the application level. All different approaches have advantages as well as disadvantages. Java-based nodes enable much faster simulations but do not run deployable code. Hence, they are useful for the development of e.g. distributed algorithms. Emulating nodes allows control and retrieval of more fine-grained execution details compared to Java-based nodes or nodes running native code. Finally, native code simulations are more efficient than node emulations and still simulate deployable code. Combining the different levels in the same simulation can give both an efficient simulation as well as fine-grained execution details on selected nodes.

2.3 Contiki

Contiki [4] is a sensor network operating system. Contiki supports three communication stacks: Rime [5], a lightweight layered communication stack that provides basic communication primitives on top of which more complex protocols are built, uIP [3] is a fully RFC compliant TCP/IPv4 stack for memory constrained systems, and uIPv6 [9], the world's smallest fully RFC compliant TCP/IPv6 stack.

Contiki has an on-line power profiling mechanism [7] which estimates the energy consumption by measuring the duration each component is in various modes such as low-power mode, transmitting.

2.4 TinyOS

TinyOS is a sensor network operating system popular in academia originally targeting hardware with just 512 Bytes of RAM [14]. The main difference to other sensor network operating systems is the use of the specifically developed programming language nesC [12] that builds component abstractions on top of standard C. This programming language has to be used by application developers, since the application together with system components are used to generate a single binary image to be programmed to the sensor nodes. TinyOS is event-based, which is supported by special constructs in nesC, and requires also the application developer to follow this programming model.

The basic communication abstraction of TinyOS is a simple best-effort one-hop message transmission service. In addition to the frame format of the data link layer used by the radio chip, e.g., IEEE 802.15.4, TinyOS just adds one byte – the Active Message Type – to differentiate among up to 256 different software components as the intended destination on the receiver. Building on this abstraction, other protocols can be built. As an alternative to this, a partial IPv6 stack [13] has been added in recent TinyOS releases.

No mechanism for on-line estimation of power consumption is available. Instead, algorithms that require this information, e.g., for lifetime estimation and adaption of functionality, require an extensive prior evaluation by simulation to obtain estimates for inclusion in the deployment [17].

3. IMPLEMENTATION

By integrating MSPSim more closely into COOJA and adding some OS-specific support, we get a simulation tool that can simulate sensor networks consisting of both Contiki and TinyOS nodes.

3.1 Simulating TinyOS nodes

The main difference between simulating Contiki nodes and TinyOS nodes is that they use different node ID variables. Contiki uses *node_id* for its node ID while TinyOS uses the constant TOS_NODE_ID and the mutable Active Message address $TOS\_AM\_ADDRESS$. Initially, we added support in COOJA for setting just the node id, but initial experiments indicated that it is also practical to be able to set the Active Message address through COOJA.

In MSPSim, we made the emulation of the MSP430 and the CC2420 radio chip more accurate and complete. The initial emulation was limited to the subset of features used in only one of the operating systems. One important addition is the SFD capture interrupt that might be used for time stamping, etc.

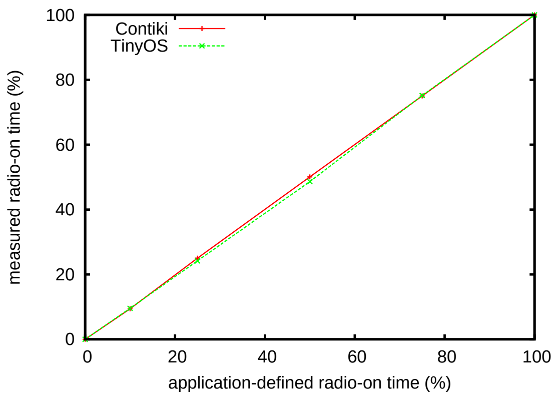


Figure 5: COOJA/MSPSim correctly measures the power consumption of both TinyOS and Contiki nodes

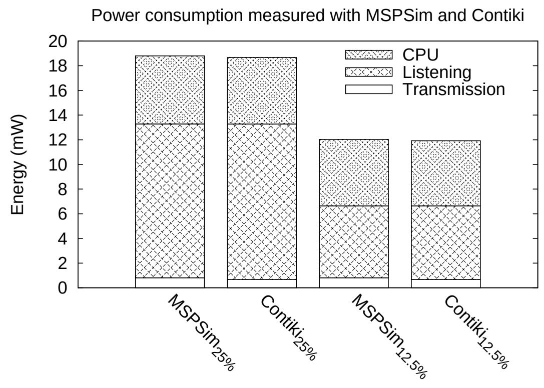


Figure 6: Energy estimation with MSPSim and Contiki for two duty cycle cases, 25% and 12.5%.

3.2 Power Profiling of all nodes

To power profile non-Contiki firmwares in COOJA we also extended MSPSim with more detailed statistics for external components such as the CC2420 radio chip. The mechanism is similar to the built-in power profiling mechanism in Contiki. The information can be accessed per node from COOJA when power profiling is needed.

4. EVALUATION

In this section we present results from experiments with the COOJA/MSPSim simulator.

4.1 Measuring Power Consumption

To evaluate whether the radio duty cycle measurement in MSPSim works as expected for both Contiki and TinyOS nodes, we have written TinyOS and Contiki applications that turn off the radio for 10%, 25%, 50%, 75% and 100% of the time. We show that the simulator is able to measure the radio duty cycle with the corresponding values.

Our results depicted in Figure 5 show that the differences between the expected and the measured values are very small for both operating systems, namely at maximum around 1%.

4.2 Measuring Power Consumption with MSPSim

To evaluate the accuracy of MSPSim's built-in power consumption measurements we compare with the energy consumption estimation provided by Contiki's power profiling.

The test application sends data packets at a regular interval and toggles the radio transceiver on and off with a given duty cycle.

We measure the energy consumption in MSPSim by printing the duty cycle of CPU active and the CC2420 radio listen and transmit modes as shown below:

```
>duty 1 "MSP430 Core.active" CC2420
10.39 74.99 23.48 1.53
10.39 74.99 23.48 1.53
10.39 74.99 23.48 1.53
...
```

The columns in the output are percentages of CPU activity and the modes of the CC2420 radio: power down, listen, and transmit. We calculate the energy consumption by multiplying the time spent in each mode with the respective power consumption in milliwatts.

The results in Figure 6 show that the energy estimations are very close to each other in both duty cycle cases. The maximum difference is around two percent on listen and CPU, but we observe a somewhat larger difference when estimating the power consumption of transmissions. The reason for this difference is that the built-in measurements in MSPSim have immediate knowledge when the radio switches to transmission mode, while the Contiki counterpart must read status registers and therefore gets delayed information about when transmission starts.

4.3 A Heterogeneous, Hierarchical Sensor Network

To demonstrate interoperability between Contiki and TinyOS in COOJA/MSPSim, we simulate a heterogeneous, hierarchical sensor network shown in Figure 1. The hierarchical sensornet consists of both TinyOS and Contiki nodes. The three TinyOS nodes placed in the right part of the top right plug-in in Figure 1 act as data sources that periodically send a data message with 20 bytes payload ("As", "Bs" or "Cs") to an aggregator node running Contiki. The aggregator node aggregates the data and sends it via a forwarder node in a multi-hop fashion to the sink (left-hand node in the plug-in). Both the forwarder and the sink node run Contiki.

Since the TinyOS frame format differs from the Contiki frame format, we have modified the Rime stack and the Chameleon module [5] to allow Contiki to understand packets from the TinyOS nodes.

In our experiments we send 200 packets from each TinyOS node. The aggregator node reduces the number of packets from 600 to 200. If this reduction of the number of packets leads to energy savings depends to a large extent on the MAC layer. In order to quantify the power savings, we have measured the power consumption with and without aggregation over two hops, i.e. from the aggregator node via the forwarder to the sink node. In these results, the aggregator either sends one large packets with 60 Bytes payload every three seconds or three shorter packets with 20 Bytes payload every second.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5637
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5637

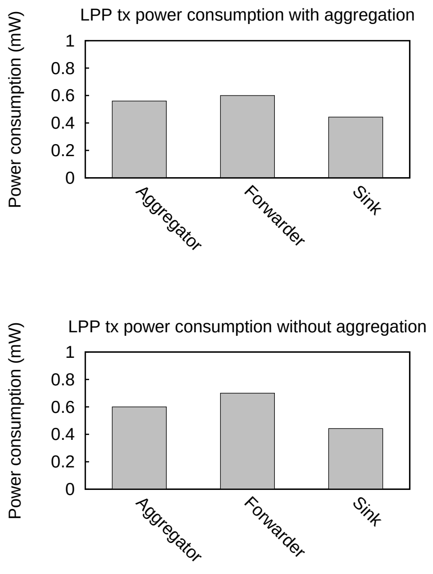


Figure 7: Aggregation reduces TX power consumption.

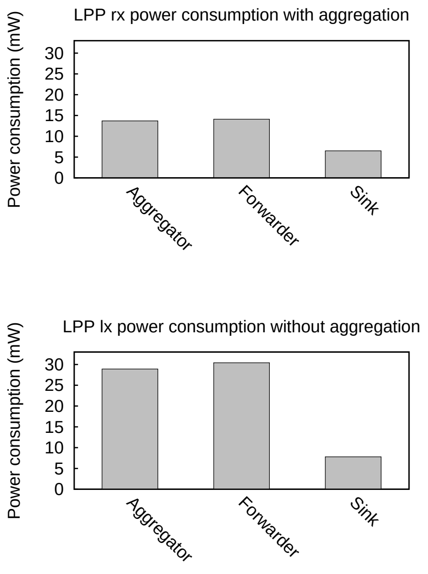


Figure 8: Aggregation reduces RX power consumption

We use low power probing (LPP) as the underlying MAC layer [20]. LPP receivers periodically transmit probes. Essentially, probes are short packets that announce that the node in question is awake and ready to receive a data packet. After sending a probe, receivers keep their radio on for a short time to listen for data packets. A sender that has a packet to be sent turns on its radio waiting for a probe from a neighbor it wants to send to. On the reception of a probe from a potential receiver, the node sends a small hardware ACK that is followed by the actual data packet. We have used the default LPP configuration available in Contiki that sends on average four probes per second and that keeps the radio on for $\frac{1}{128}$ seconds after each probe.

Our results are shown in Figure 7 and Figure 8. Figure 7 shows that the reduction of the number of packets also leads to reduced power consumption for transmitting packets (TX power consumption) for the forwarder and the aggregator nodes. The TX power consumption for the sink nodes is not influenced which indicates that the reduced power consumption is not caused by a change of the number probing packets but by the reduced number of header bytes that needs to be transmitted.

As expected, the power consumption for the radio in listen mode (RX power consumption) is much lower than the TX power consumption. Using LPP, a node that wants to transmit a packet needs to turn the radio on and keep it in listening mode until it receives a probe from the receiver. Without aggregation, the number of packets a node transmits increases and hence the time a transmitter needs to keep the radio on. The results in Figure 8 show that the power consumption more than doubles without aggregation.

4.4 Interoperability Tests

To validate our simulation tool's capability for interoperability tests we perform an experiment where we develop the same networked application in both TinyOS and Contiki.

We implement an application that broadcasts packets to its neighbors. When it receives packets it counts the numner of neighbors and shows the neighbor count on the leds. The application for TinyOS uses the standard Active Message communication framework and the Contiki application is designed to replicate the TinyOS application. Since TinyOS uses IEEE 802.15.4 we replace the default Contiki MAC protocol with the IEEE 802.15.4 MAC and add the TinyOS active message type as the first byte of the payload.

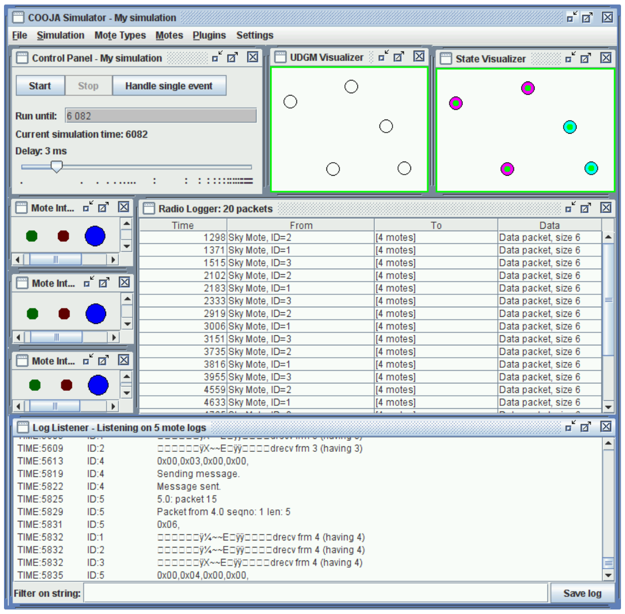
Figure 9 shows the result of the test. All nodes communicated and counted neighbors as expected.

This interoperability test validates a very basic application protocol on top of 802.15.4 but still shows that it is possible to perform interoperability tests using COOJA/MSPSim.

5. RELATED WORK

During recent years, a number of wireless sensor networking simulators have been developed. Most of these cannot be used for interoperability testing. Many simulators are developed for specific operating systems. An example is the TOSSIM simulator [18] that only simulates nodes running the TinyOS operating system [14]. These simulators usually run the same application code, but it is compiled for the simulator's host and not directly deployable without recompilation for the sensor device hardware. Other simulators such as Castalia [1] and MiXiM [11] do not simulate the operating system and the application code, but simulate at a higher level and with a focus on network related aspects.

Instruction level simulators such as MSPSim and Avrora [24] are able to simulate nodes running different operating systems since they operate at the instruction set level. Avrora emulates Mica2 sensor nodes and can emulate sev-


COOJA Simulator - My simulation
File Simulation Mote Types Motes Plugins Settings
Control Panel - My simulation
Start Stop Handle single event
UDGM Visualizer
State Visualizer
Mote Int...
Mote Int...
Mote Int...

Run until: 6 082
Current simulation time: 6082
Delay: 3 ms

Radio Logger: 20 packets

Time	From	To	Data
1298	Sky Mote, ID=2	[4 motes]	Data packet, size 6
1371	Sky Mote, ID=1	[4 motes]	Data packet, size 6
1515	Sky Mote, ID=3	[4 motes]	Data packet, size 6
2102	Sky Mote, ID=2	[4 motes]	Data packet, size 6
2183	Sky Mote, ID=1	[4 motes]	Data packet, size 6
2333	Sky Mote, ID=3	[4 motes]	Data packet, size 6
2919	Sky Mote, ID=2	[4 motes]	Data packet, size 6
3006	Sky Mote, ID=1	[4 motes]	Data packet, size 6
3151	Sky Mote, ID=3	[4 motes]	Data packet, size 6
3735	Sky Mote, ID=2	[4 motes]	Data packet, size 6
3816	Sky Mote, ID=1	[4 motes]	Data packet, size 6
3955	Sky Mote, ID=3	[4 motes]	Data packet, size 6
4559	Sky Mote, ID=2	[4 motes]	Data packet, size 6
4633	Sky Mote, ID=1	[4 motes]	Data packet, size 6

Log Listener - Listening on 5 mote logs

```
TIME:5609   ID:2     □□□□□□□ÿX~~E□ÿÿ□□□□drecv frm 3 (having 3)
TIME:5613   ID:4     0x00,0x03,0x00,0x00,
TIME:5819   ID:4     Sending message.
TIME:5822   ID:4     Message sent.
TIME:5825   ID:5     5.0: packet 15
TIME:5829   ID:5     Packet from 4.0 seqno: 1 len: 5
TIME:5831   ID:5     0x06,
TIME:5832   ID:1     □□□□□□□ÿ¼~~E□ÿÿ□□□□drecv frm 4 (having 4)
TIME:5832   ID:2     □□□□□□□ÿ¼~~E□ÿÿ□□□□drecv frm 4 (having 4)
TIME:5832   ID:3     □□□□□□□ÿX~~E□ÿÿ□□□□drecv frm 4 (having 4)
TIME:5835   ID:5     0x00,0x04,0x00,0x00,
```

Filter on string: Save log

Figure 9: Basic interoperability test with three TinyOS nodes and two Contiki nodes communicating. Some of the nodes' led-panels are shown, and debug printouts are visible in the log window.

eral nodes simultaneously. It supports communication by emulating the Chipcon CC1000 radio chip. AvroraZ [2] is an extension of Avrora that provides a detailed emulation of the Texas Instruments Chipcon CC2420 radio chip including an indoor radio model. Avrora and AvroraZ run multiple nodes using Java threads while our simulation tool COOJA/MSPSim schedules the nodes explicitly. In contrast to these efforts, we have shown that COOJA/MSPSim can simulate nodes with different operating systems in the same simulation and perform power profiling.

6. CONCLUSIONS

We present a simulation-based approach for white-box interoperability testing in sensor networks. By combining simulations at the network and the hardware layer, we enable white-box testing in heterogeneous sensor network environments. Our approach makes interoperability testing more practical and transparent by allowing repeatable and fine-grained control of experiments. We demonstrate our simulation tool through experiments in which applications of two different operating systems exchange protocol data. Furthermore, we show that the new white-box testing environment allows accurate and non-intrusive power consumption measurements at the network scale.

More information about COOJA and MSPSim, including download links, can be found at:
http://www.sics.se/contiki/
and
http://www.sics.se/project/mspsim/

7. 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. This work was also partly financed by VINNOVA, the Swedish Agency for Innovation Systems.

8. REFERENCES

[1] A. Boulis. Castalia: revealing pitfalls in designing distributed algorithms in WSN. In *Poster proceedings of the 5th international conference on Embedded networked sensor systems*, pages 407–408, 2007.

[2] Rodolfo de Paz Alberola and Dirk Pesch. Avroraz: Extending avrora with an ieee 802.15.4 compliant radio chip mode. In *3rd ACM International Workshop on Performance Monitoring, Measurement, and Evaluation of Heterogeneous Wireless and Wired Networks*, Vancouver, Canada, October 2008.

[3] A. Dunkels. Full TCP/IP for 8-bit architectures. In *Proceedings of The First International Conference on Mobile Systems, Applications, and Services (MOBISYS '03)*, May 2003.

[4] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Workshop on Embedded Networked Sensors*, Tampa, Florida, USA, November 2004.

[5] A. Dunkels, F. Österlind, and Z. He. An adaptive communication architecture for wireless sensor networks. In *Proceedings of the Fifth ACM Conference on Networked Embedded Sensor Systems (SenSys 2007)*, Sydney, Australia, November 2007.

[6] 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.

[7] 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.

[8] A. Dunkels and J-P. Vasseur. IP for Smart Objects, September 2008. IPSO Alliance White Paper #1.

[9] M. Durvy, J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels. Making Sensor Networks IPv6 Ready. In *Proceedings of the Sixth ACM Conference on Networked Embedded Sensor Systems (ACM SenSys 2008)*, Raleigh, North Carolina, USA, November 2008.

[10] J. Eriksson, A. Dunkels, N. Finne, F. Österlind, and T. Voigt. MSPSim – an extensible simulator for MSP430-equipped sensor boards. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*, Delft, The Netherlands, January 2007.

[11] Köpke et al. Simulating wireless and mobile networks in omnet++: The mixim vision. In *First International OMNeT++ Developers Workshop*, March 2008.

[12] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC language: A holistic approach to networked embedded systems. In *Proceedings of the ACM SIGPLAN 2003 conference on Programming*

language design and implementation, pages 1–11, 2003.

[13] Matus Harvan and Jürgen Schönwälder. A 6lowpan implementation for TinyOS 2.0. In *Proc. of the 6th GI/ITG KuVS Fachgespräch "Wireless Sensor Networks"*, Aachen, 2007.

[14] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, November 2000.

[15] F. Kim, A.and Hekland, S. Petersen, and P. Doyle. When hart goes wireless: Understanding and implementing the wirelesshart standard. In *13th IEEE Conference on Emerging Technologies and Factory Automation*, Hamburg, Germany, September 2008.

[16] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis. Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea. In *Proceedings of the 3rd international conference on Embedded networked sensor systems (SenSyS)*, San Diego, CA, USA, November 2005.

[17] Andreas Lachenmann, Pedro José Marrón, Daniel Minder, and Kurt Rothermel. Meeting lifetime goals with energy levels. In *Proc. of the 5th ACM Conference on Embedded Networked Sensor Systems*, 2007.

[18] 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.

[19] P. Marrón, R. Sauter, O. Saukh, M. Gauger, and K. Rothermel. Challenges of complex data processing in real world sensor network deployments. In *Proceedings of ACM Workshop on Real-World Wireless Sensor Networks (REALWSN'06)*, Uppsala, Sweden, June 2006.

[20] R. Musaloiu-E., C-J. M. Liang, and A. Terzis. Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks. In *IPSN '08*, 2008.

[21] 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, November 2006.

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

[23] 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.

[24] B.L. Titzer, D.K. 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)*, April 2005.

[25] T. Voigt, F. Österlind, N. Finne, N. Tsiftes, Z. He, J. Eriksson, A. Dunkels, U. Båmsted, J. Schiller, and K. Hjort. Sensor networking in aquatic environments – experiences and new challenges. In *Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2007)*, Dublin, Ireland, October 2007.

[26] Zigbee. Web page. 2007-11-21. http://www.zigbee.org.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5637
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5637