Demo Abstract: Debugging Wireless Sensor Network Simulations with YETI and COOJA

Richard Huber, Philipp Sommer, and Roger Wattenhofer
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
{rihuber,sommer, wattenhofer}@tik.ee.ethz.ch

ABSTRACT

Developing applications and protocols for wireless sensor networks is a time consuming and error prone task, which possibly includes many iterations of implementation and testing. While simulations will probably not replace testbed experiments completely, they are a time efficient tool for rapid prototyping of sensor networks. Writing code and testing an application is often done using a different set of tools, e.g., a text editor and a simulation tool. To bridge the gap between development and simulation, we demonstrate the advantages of a seamless interconnection between YETI, a feature-rich development environment for TinyOS, and COOJA, a simulator for wireless sensor networks.

Categories and Subject Descriptors

D.2.5 [**Testing and Debugging**]: Testing tools

General Terms

Design, Experimentation

Keywords

Sensor Networks, TinyOS, Debugging, Development, Simulation

1. INTRODUCTION

Developing and debugging applications for wireless sensor networks is know to be cumbersome and time consuming. To address the challenges imposed by the embedded nature of sensor nodes and fluctuating environmental conditions, various hardware and software tools have been developed to facilitate the debugging process. Debugging tools such as a JTAG hardware adapter can be used to monitor a single attached node. On the other hand, testbeds consisting of dozens to hundreds of nodes allow to test and monitor large scale applications under realistic settings. Simulation tools are widely used in the initial phase of development. They make the execution of a program repeatable, and offer an easy way to explore the parameter space of the implementation in a controlled environment. Therefore, several simulation tools have been developed by the sensor network community, each emphasizing on a different aspect of wireless sensor network research. Castalia [1] is mainly targeted towards the realistic simulation of the wireless channel. TOSSIM [4] simulates TinyOS applications by replacing low-level components of the nodes with its own implementation in order to balance between scalability and fidelity. Both MSPSim/COOJA [3] (MSP430 platform) and Avrora [5] (AVR platform) provide cycle-accurate simulations at the hardware level.

IPSN'11, April 12–14, 2011, Chicago, Illinois.

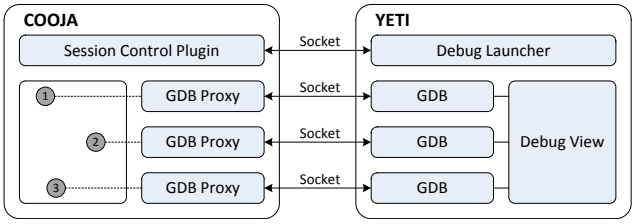

Figure 1: Interconnecting COOJA and YETI: Both tools run as independent processes. They are connected using TCP sockets to exchange metadata and control the debugging process.

2. CONNECTING COOJA AND YETI

We demonstrate how developers can benefit from the seamless interconnection of YETI, an integrated development environment for TinyOS, and the CCOJA/MSPSim network simulator. Our approach offers TinyOS developers a graphical solution to test their developed applications by simulation, without the hassle of command line operations.

TinyOS Development. YETI [2] is an Eclipse plug-in for TinyOS development, providing syntax highlighting, code completion, and error detection.[1] It is built around a sophisticated parser for the nesC language and integrates seamlessly with any existing TinyOS tool chain.

WSN Simulation. The COOJA network simulator [3] is a simulation environment for wireless sensor networks. It uses MSPSim, a low-level emulator for MSP430 microcontrollers. Even though MSPSim and COOJA have been developed for

[1]YETI is available from the project repository at http://code.google.com/a/eclipselabs.org/p/yeti/

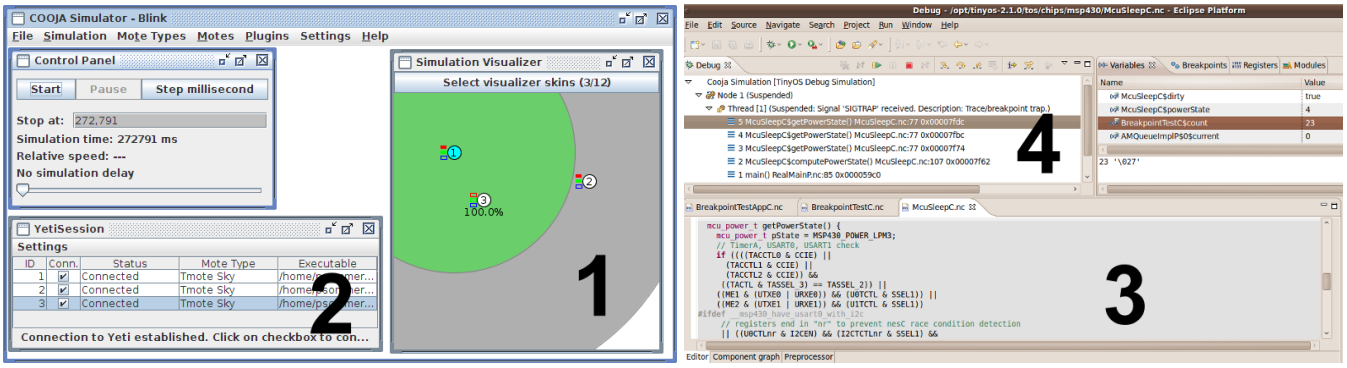

Figure 2: Basic work flow during a debug session: First, the simulation is configured in COOJA (1), and the session control plug-in is started (2). Then, breakpoints can be added or removed in YETI's nesC editor (3). When the simulation is stopped at a breakpoint, the user can step through the code in YETI (4).

the Contiki operating system, they can also be used for the simulation of TinyOS applications, since the program code is emulated at the instruction level. COOJA and MSPSim can be extended through a flexible plug-in architecture, which allows to monitor and control almost every aspect of the network and the simulated nodes.

System Architecture. By connecting simulations running in COOJA with the debugging capabilities of YETI, we can provide a powerful development and debugging environment to the user. The software architecture of our approach is depicted in Figure 1. Even though both tools are based on Java, we decided in favor of a loosely coupling of the two tools using TCP sockets, rather than integrating one tool into the other.

GDB remote protocol. The GNU debugger (GDB) contains debugging support for remote targets using a simple command and response based protocol. Debugging *commands* are sent by the host (GDB) to the remote target, which answers with a corresponding *response* packet. We implemented a remote debugging stub for GDB as a plug-in for MSPSim/COOJA, which allows GDB to connect to a simulated node using a TCP connection to a specific port. This allows to set breakpoints in MSPSim through GDB, and to read and/or modify the content of the node's register and memory space. GDB itself is already tightly integrated into Eclipse's C Development Tools (CDT), which are used to provide debugging support with YETI.

Session control. Originally, GDB has been designed for debugging a single process at a time only. Although recent development of GDB improved support for debugging multiple threads or processes, we decided to use a dedicated socket connection to exchange information about the current simulation between YETI and COOJA. Therefore, when the user starts a new debugging session in Eclipse, a connection to the session control plug-in on the COOJA side is established, and YETI requests information about the running simulation, e.g., the number of nodes and the paths to the corresponding binary images. Furthermore, YETI queries the session control plug-in for the TCP port at which the corresponding GDB stub is listening for incoming debug sessions. This allows the user to add and remove nodes to/from a running debugging session on the fly. Each node in the simulation is handled by a new GDB process within Eclipse.

3. USER INTERACTION

Connecting a TinyOS project in YETI with a COOJA simulation for debugging purposes includes four simple steps, as shown in Figure 2.

Simulation setup. In a first step, the simulation scenario has to be configured in COOJA. This includes the setup of the binary images executed by the simulated nodes and the assignment of node positions. Furthermore, general simulation parameters such as the communication range or the packet loss may be adjusted.

Debug session. When launching a new debug session, YETI spawns a new GDB process for each connected node in the simulation. Building upon the familiar functionality of Eclipse's integrated debugging tools and views, the current status and stack trace of each simulated node is shown.

Breakpoints. Inserting or removing breakpoints can be done on a per node basis in YETI's source code view of the corresponding TinyOS module. The address of a breakpoint is then sent to the corresponding GDB stub, which registers the breakpoints in the COOJA simulator. Every time a breakpoint is reached by a node, the simulation is stopped and the content of the registers and the memory is updated in YETI's debugging view.

Read/modify node state. Register and memory of all nodes being part of the simulation can be inspected and modified as long as the simulation is paused, e.g., when a node has reached a breakpoint. Changes to the value of variables or registers made by the user are written back to the COOJA simulation immediately.

4. REFERENCES

[1] A. Boulis. Castalia: Revealing Pitfalls in Designing Distributed Algorithms in WSN. In *SenSys*, 2007.

[2] N. Burri, R. Flury, S. Nellen, B. Sigg, P. Sommer, and R. Wattenhofer. YETI: An Eclipse Plug-in for TinyOS 2.1. In *SenSys*, 2009.

[3] 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 *Simutools*, 2009.

[4] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *SenSys*, 2003.

[5] B. L. Titzer, D. K. Lee, and J. Palsberg. Avrora: Scalable Sensor Network Simulation with Precise Timing. In *IPSN*, 2005.