Demonstration of worldsens: a fast prototyping and performance evaluation of wireless sensor network applications protocols

Guillaume Chelius, Antoine Fraboulet, Eric Fleury

Demonstration of Worldsens: A Fast Prototyping and Performance Evaluation of Wireless Sensor Network Applications & Protocols

Guillaume Chelius
INRIA/ARES
CITI/INSA de Lyon, France
guillaume.chelius@inria.fr

Antoine Fraboulet
CITI/INSA de Lyon
INRIA/Compsys, France
antoine.fraboulet@insa-lyon.fr

Eric Fleury
CITI/INSA de Lyon
INRIA/ARES, France
eric.fleury@insa-lyon.fr

ABSTRACT

We present Worldsens, a complete environment for fast prototyping of wireless sensor protocols and applications. Our environment proposes a full simulation platform with both embedded software instruction and radio packet accuracy. We propose a demonstration including a full software design, simulation, performance estimation and deployment on a set of nodes within the same design environment. Through these first experimentations, we show that accurate sensor network simulation is feasible and that complex application design and deployment is affordable.

Categories and Subject Descriptors

D.2.5 [**Software Engineering**]: Testing and Debugging—*Debugging aids, Monitors*; D.2.6 [**Software Engineering**]: Programming Environments—*Integrated environments*

General Terms

Design, Experimentation, Performance

Keywords

Wireless Sensors, Application Design, Simulation Tools

1. WORLDSENS PROPOSAL

The aim of this demo is to demonstrate the advantages of Worldsens, an integrated platform that enables the design, development, prototyping and deployment of applications on wireless sensor networks. We do believe that offering the "right" simulation tool is a key feature in the development of sensor application development and more generally for the sensor research community. Sensor network activities are preponderant nowadays and address a large spectrum of research topics: architecture, OS, network protocols, energy power saving, data aggregation, browsing and processing, distributed computing...

It is quite clear that in such challenging domain, very related to the real world, we do need simulation tools that address the sensor network application in its globality and not only part of it. Moreover, in order to validate the application, our goal is to provide a simulation platform that handles the real application native code in order to be able to test and validate the application at the *instruction* level. We do not want to add a burden to the programmer by developing a new model of his application, or by transforming (automatically or not) low level part as sensor network application may reside in the low level part. For example, `ns-2` is useful to study an abstracted behavior of a specific part but it does not provide a way to study an implementation of algorithms running at the level of the real application.

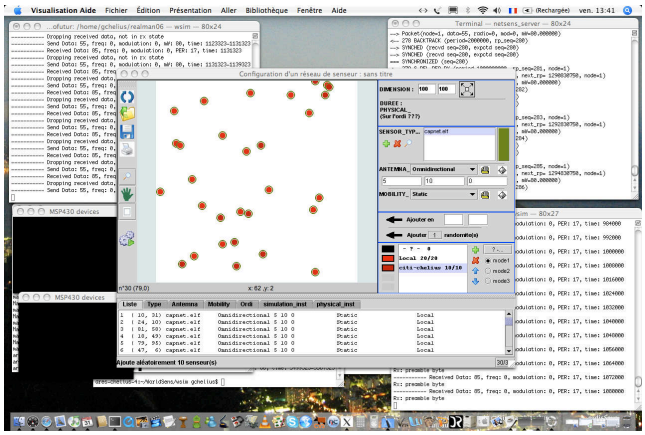

Figure 1: User Interface used to manage sensor network simulators.

Our simulator was used to build a complete sensor network using 200 mobile nodes and several base stations. Using the simulation environment we were able to develop all the embedded software and base station data collection application before the first deployment. The demonstration we propose will set up the complete software realization from the ground up to a sensor network deployed on real hardware used to build interaction graphs.

In order to handle such a challenging application, we have designed our Worldsens simulation tool with respect to the following goals

accuracy: Worldsens captures the behavior of the sensor node application at a very low level, using the native code: instruction level, bit level and interrupt level.

tunable network model: Worldsens reflects the behavior of the network. At the same time, one may tune the

REALMAN'06, May 26, 2006, Florence, Italy.
ACM 1-59593-360-3/06/0005.

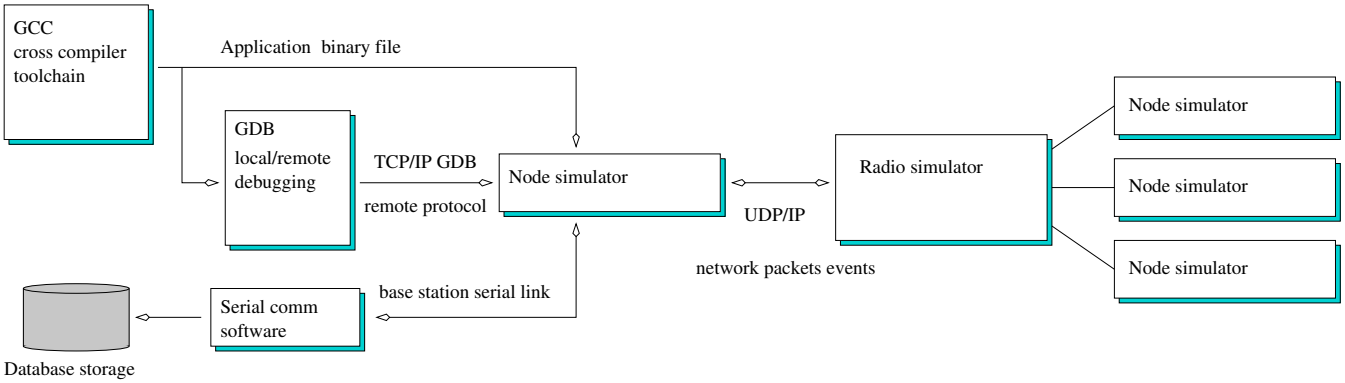

Figure 2: Distributed simulation environment for Wireless Sensor Networks.

degree of accuracy from an ideal network layer where all transmitted packet are received to a network model taking into account collisions, SNR and radio propagation.

Scalability: In order to model the behavior of a global application, Worldsens handles large sensor networks, with hundred to thousand nodes. The first application developped using Worldsens is composed of 200 nodes.

Time: Worldsens handles precise timing in order to get the behavior of interruptions and bit level radio simulation.

To resume the several functionalities of our platform let us describe briefly the components of our architecture. The global architecture of our simulation platform is depicted in Figure 2.

The central component is the *Node simulator* that takes as input the binary file of the application. The simulator runs the native code as deployed in the sensor hardware without any change. In order to do this, the node simulator emulates all components embedded in the hardware sensor nodes: flash memory module, DS2411 1-Wire silicon serial number, CC1100 multi-channel RF transceiver as depicted on Figure 3. Thus, all instructions sending commands to the CC1100 are executed and the behavior of the CC1100 is also simulated. Each node simulator is also connected to a central *radio simulator* which simulates radio propagation and interference, thus impacting the way other CC1100 RF transceiver receives bits. Finally one can connect a remote gdb instance on each node simulator, allowing a step by step execution at the instruction level (C level and/or assembler level). Note that the serial port is also available inside the Worldsens platform in order to simulate the communication between a sensor node and a PC/base station for example.

2. PLATFORM DESCRIPTION

2.1 Hardware Simulation

Our development environment relies on cycle accurate full platform simulation using microprocessor instruction driven timings. The simulator is able to perform a full simulation of hardware events that occur in the platform and to give back to the developer a precise timing analysis of the simulated software.

The hardware we simulate is currently based on the MSP430 micro-controller (MCU) series from Texas Instruments. The Ti MSP430 is a 16 bits MCU that includes several on-board peripherals (General Purpose I/O ports, timers, serial ports) and 64KB mixed address space for Flash and RAM memories in the same packaging. This MCU is specifically designed for low power and is able to run a full operating system.

We currently provide connectors for universal synchronous / asynchronous receive transmit (USART), SPI (Serial Peripheral Interface) or direct MSP430 general purpose I/O peripherals. These connectors are used to connect to system peripherals included in the sensor node.

Peripherals (external Flash, radio chipset, ...) are simulated using software connexions to the MCU input/output ports. We use function call-backs to simulate all interaction between the memory mapped MCU I/O and the peripherals software emulation code. These call-backs are able to generate interrupt requests to the MCU.

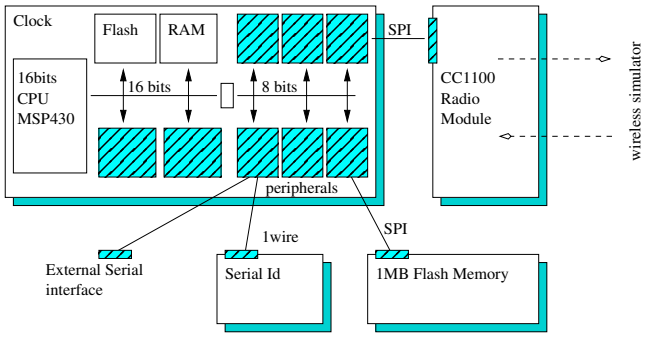

Figure 3: Wsim: wireless sensor node architecture simulator.

The software part of the node can be used in the simulator without the need to reconfigure or recompile the software. We use a classical GCC cross-compiler toolchain [3] (see Fig. 4) and the simulation is not attached to any particular language nor operating system. We are thus able to debug and evaluate performances of the full system at the assembly level. A precise estimation of timings, memory consumption and power can be obtained during simulation. FreeRTOS, Contiki and TinyOS operating systems have been successfully tested on the simulation platform.

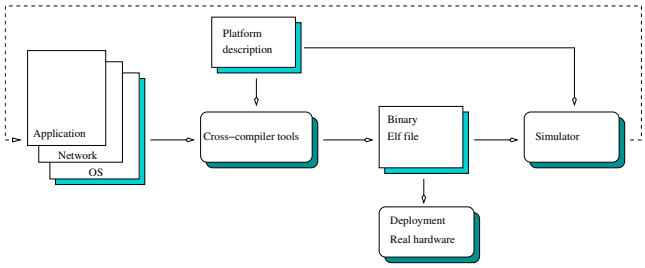

Figure 4: Application development methodology.

Our platform simulation also provides an interface for remote debugging using the `GDB` serial remote protocol (Figure 5). This allows to perform precise instruction debugging using conventionnal compiler tools. The debugging is similar to a remote debugger using a `JTAG` monitoring technique. The overall simulation speed allows us to run multiple network nodes in real time on the same machine.

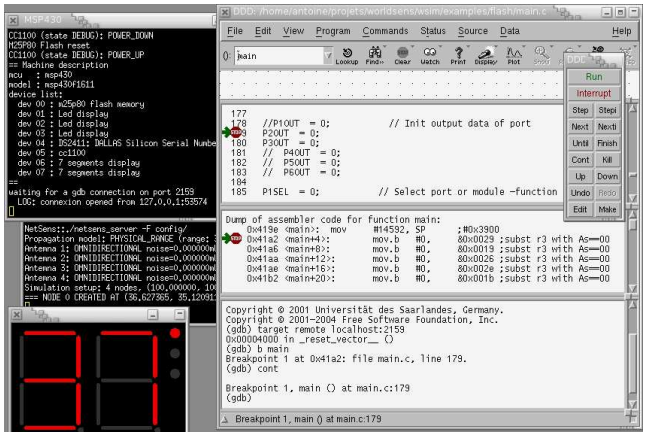

Figure 5: Wsim remote debugging.

2.2 Wireless Network simulation

This `Chipcon` module emulates the specifications of the CC1100 communication chipset. It is also a wrapper that connects the hardware node emulator to a wireless medium simulator executed in an indepependant process. In consequence, a whole sensor network is simulated using several hardware node emulators together with the wireless medium simulator. Each node emulator executes its application, communicates using the `Chipcon` emulator, and radio communications are solved by the wireless medium simulator with a radio symbol accuracy. As inter-process communications between the simulators are UDP ones, the simulation of a sensor network can be distributed over several computers.

The network simulator is event driven whereas the architecture simulators are cycle driven. Moreover, the simulation is distributed and node simulators are executed in independant and parallel processes where no global clock is available. As a consequence, a synchronization mechanism is required in order to ensure a correct behaviour of the network regarding the execution of each node. This synchronization is based on rollbacks on deadlocks [2] imposed by the radio simulator either optimistically or upon radio events. This allows us to have a parallel simulation environment that can handle both cycle precise architecture and event driven packet simulation.

The wireless medium simulation is parameterized with several configuration files describing all the caracteristics of the physical medium and each network node. In particular, nodes are abstracted by the instantiation of three parameters:

- Antemna model (e.g. omnidirectional, directional antenas);
- Mobility model (e.g. random waypoint, static, group mobility);
- Radio model (e.g. FDMA, CDMA);

The wireless medium is also abstracted by the instantiation of two parameters:

- Propagation model (e.g. range, two-ray or $1/r^{\alpha}$);
- Interference model;

The wireless network simulator considers all these parameters to compute a SNR (*Signal over Noise Ratio*) and BER (*Bit Error Rate*) for each radio symbol sent over the medium and for each receiving node. Modulation models are used to deduce the BER from the computed SNR, radio resources and interference models are used to compute interferences betwen different radio packets as well as to deduce the SNR and finally antena and propagation models are used to compute the strength of a signal at a given location..

A great advantage of this architecture and its model instantiation/configuration is that a same application can be simulated over different radio, antenas or physical models simply by changing the configuration files. It becomes easier to study the impact of the physical, antenas and radio layers on the protocol performances.

3. REFERENCES

[1] Lewis Girod et al. A system for simulation, emulation, and deployment of heterogeneous sensor networks. In *SenSys'04*. ACM Press, November 2004.

[2] Alois Ferscha. *Parallel and Distributed Computing Handbook*, pages 1003 – 1041. McGraw-Hill, 1996.

[3] Gcc port for msp430. web, `http://mspgcc.sourceforge.net/`, November 2005.

[4] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *SenSys'03*. ACM Press, November 2003.