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

Guillaume Chelius, Antoine Fraboulet, Eric Fleury

To cite this version:

HAL Id: hal-00399616
https://hal.science/hal-00399616
Submitted on 27 Jun 2009

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 network 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.

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
Figure 2: Distributed simulation environment for Wireless Sensor Networks.

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 connections 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.

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.

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.
Our platform simulation also provides an interface for remote debugging using the GDB serial remote protocol (Figure 5). This allows us to perform precise instruction debugging using conventional 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.

The network simulator is event driven whereas the architecture simulators are cycle driven. Moreover, the simulation is distributed and node simulators are executed in independent 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 characteristics of the physical medium and each network node. In particular, nodes are abstracted by the instantiation of three parameters:

- Antenna model (e.g. omnidirectional, directional antennas);
- 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 $1/r^x$);
- Interference model;

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

3. REFERENCES


