



Demo Abstract: Distributed and Synchronized Measurements with FLOCKLAB

Roman Lim, Christoph Walser, Federico Ferrari, Marco Zimmerling, Jan Beutel
Computer Engineering and Networks Lab
ETH Zurich, Switzerland
lim@tik.ee.ethz.ch

1 Motivation and Overview

Developing, testing, debugging, and evaluating communication protocols for low-power wireless networks is a long and cumbersome task. Simulators can be helpful in the early stages of development, but their models of hardware components and the wireless channel are often rather simplistic and hence cannot substitute experiments on real sensor node platforms. The resources available on common platforms are however very limited, and so are the possibilities for non-intrusive debugging and testing. With most existing testbeds it is only possible to collect information from the serial port, which requires adding highly intrusive logging statements that alter the timing behavior of the software running on the nodes. This is particularly detrimental to the operation of time-critical components, such as radio drivers, media access control (MAC) protocols, and certain flooding protocols [2], hindering their testbed-assisted development.

To overcome the limitations of current testbeds, we have been developing the FLOCKLAB testbed over the past years. This demo abstract describes the architecture of FLOCKLAB and highlights the benefits it brings for non-intrusive testing and profiling of sensor network protocols and applications. For example, it is possible to take synchronized power measurements and capture changes in the state of General Purpose Input/Output (GPIO) pins on every FLOCKLAB node simultaneously. The FLOCKLAB architecture is highly scalable and allows for easy integration of future node platforms. As FLOCKLAB is publicly accessible, the demonstration will provide a hands-on experience for potential testbed users.

2 The FLOCKLAB Testbed

FLOCKLAB consists of several *observer* nodes (currently 30) that are connected to a server through a wired or wireless local area network (LAN). The observer nodes are deployed both indoors and outdoors of an ETH office building. Compared with a typical sensor node, a FLOCKLAB observer is more powerful: it is based on a Gumstix Verdex pro XL6P that features a 600MHz XScale processor, 128 MB SDRAM, 32 MB flash, and runs OpenEmbedded Linux.

Sensor nodes, called *targets*, are attached to the observers through target adapter boards; up to four targets can be attached to a single observer, as shown in Fig. 1. This approach

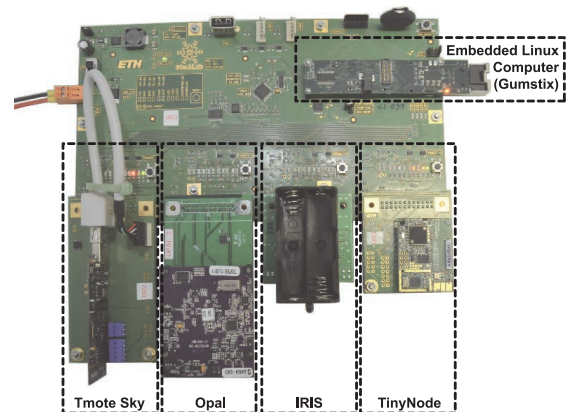


Figure 1. A FLOCKLAB observer node with four target sensor nodes attached through target adapter boards.

provides maximum flexibility as it makes FLOCKLAB support multiple types of target node architectures, selectable in software to create complex, heterogeneous testing scenarios. FLOCKLAB currently has four target adapter boards for Tmote Sky, Tynode, IRIS, and Opal nodes. Nevertheless, our design makes it easy to add other target nodes by simply designing corresponding adapter boards.

2.1 Services

The FLOCKLAB observer nodes provide hardware and software support for minimally intrusive, yet powerful monitoring and stimulation of the target sensor nodes. As shown in Fig. 2, this comprises the following four services:

- **GPIO monitoring:** An observer node can monitor up to five GPIO pins of a target node at a frequency of up to 10kHz, providing virtually non-intrusive tracing of state changes in the running program. The monitoring service can be used, for instance, to observe changes between the different radio states by setting and clearing GPIO pins at the corresponding location in the radio driver, thus allowing for accurate, non-intrusive duty cycle measurements.
- **GPIO setting:** An observer can also set a GPIO pin of a target node at a given time. This service can be used, for example, to trigger some action (e.g., turn on the radio, freeze a state variable) simultaneously on all target nodes.
- **Power profiling:** This service samples the current draw of a target node at a frequency of up to 56 k samples per

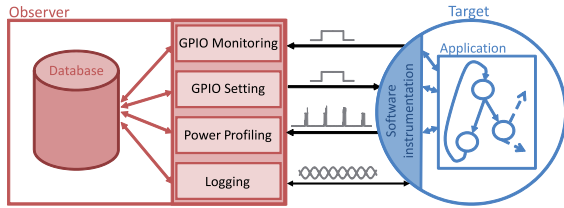


Figure 2. Services available on a FLOCKLAB observer.

second. A 24-bit analog-to-digital converter (ADC) converts the measurements into digital format.

- **Serial reader:** This standard service captures the serial output of a target node, which can be TinyOS or Contiki data packets or raw ASCII data.

All four FLOCKLAB services can potentially be active simultaneously on all observers. During a test, the data generated by the active services are first timestamped and then stored in local databases on the observers. When a test finishes, the observers upload the data to a central database on the FLOCKLAB server from where it is retrieved by the user.

2.2 User Interface

FLOCKLAB provides an easy-to-use web front-end available online at <https://www.flocklab.ethz.ch>. After registering, a user can upload test configurations and schedule test runs. A graphical overview shows the current testbed status and a connectivity map for every attached target platform, based on daily link and noise measurements.

3 Demonstration Highlights

To demonstrate the FLOCKLAB testbed, we use a small replica of the real testbed. We install three FLOCKLAB observers connected to a server instance. This allows visitors to become familiar with the web interface and to get a feel for the interactions among the different hardware components. We provide a set of sample images and test configurations, together with visualization support for the gathered test data.

As an example of performance testing with FLOCKLAB, we instrument the Collection Tree Protocol (CTP) [3] running on top of the Low-Power Listening (LPL) [6] link layer as available in the TinyOS distribution. Standard metrics to evaluate the performance of low-power communication protocols are *data yield*, *radio duty cycle*, *end-to-end packet delay* and *power draw*. With FLOCKLAB, it becomes much simpler and less intrusive to instrument, for example, CTP and LPL for such measurements compared to most existing testbeds. We demonstrate these unique benefits using Tmote Sky and Tinynode as target nodes, obtaining power measurements and GPIO traces as shown in Fig. 3:

To measure data yield, we use the serial reader service on the sink, logging the sequence numbers of received packets.

To measure radio duty cycle, we use the GPIO monitoring service and simply export the current radio state over a GPIO line. This results in instrumentation costs in the radio driver of just one instruction per state change, whereas current solutions typically need to manage several counters to measure in software the time spent in sleep and active mode.

To measure end-to-end packet delay, the distributed and synchronized GPIO monitoring feature comes in handy as

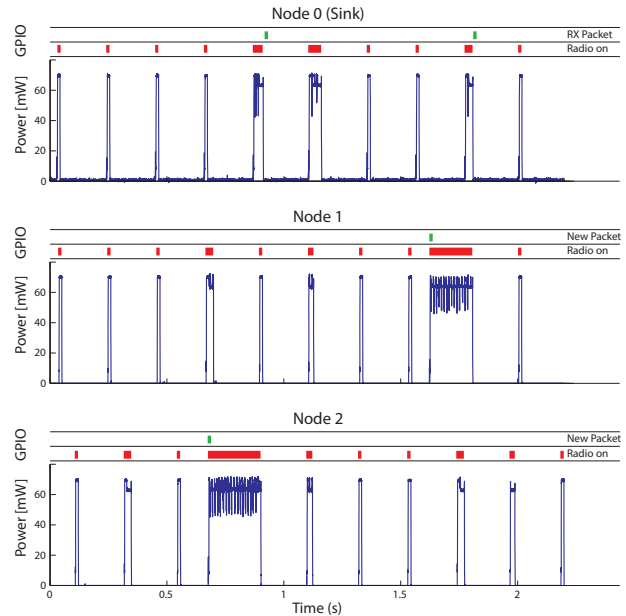


Figure 3. Power and GPIO traces taken simultaneously at three target nodes running CTP over LPL. The GPIO lines show the radio state and the time when a packet is generated at node 1 and 2, and received by node 0 (sink).

well. A node that generates a packet uses a GPIO pin to indicate the generation time, and the sink does the same to indicate the reception time. On common testbeds, users would have to instrument their code with counters to accumulate the packet lifetime in the network on every intermediate node, or use a dedicated time synchronization protocol like FTSP [5] to capture the global generation/reception time of a packet on the target nodes. The first approach tends to lack accuracy due to differing node clocks, and the second one depends on an additional protocol that induces traffic overhead.

To measure power, we activate the power profiling service on the observers. In recent years, accurate power measurements have found their way into sensor network testbeds [4]. For testbeds without this support, a feasible but less accurate approach is to track the state changes of consumers in software and use this information to compute the average power draw [1]. FLOCKLAB provides accurate and synchronized power measurements without altering the target nodes' code.

Acknowledgments. This work was supported by nano-tera.ch and NCCR-MICS under SNSF grant number 5005-67322.

4 References

- [1] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *EmNets*, 2007.
- [2] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient network flooding and time synchronization with Glossy. In *ACM/IEEE IPSN*, 2011.
- [3] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *ACM SenSys*, 2009.
- [4] I. Haratcherev, G. Halkes, T. Parker, O. Visser, and K. Langendoen. Power-Bench: A scalable testbed infrastructure for benchmarking power consumption. In *IWSNE*, 2008.
- [5] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi. The flooding time synchronization protocol. In *ACM SenSys*, 2004.
- [6] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *ACM SenSys*, 2004.