Sensor Network Experimentation using TARWIS

Philipp Hurni, Markus Anwander, Gerald Wagenknecht, Thomas Staub, Torsten Braun
Institute of Computer Science and Applied Mathematics
University of Bern, Neubrückstrasse 10, 3012 Bern, Switzerland
{hurni, anwander, wagen, staub, braun}@iam.unibe.ch

***Abstract*—Research in the area of Wireless Sensor Networks (WSNs) has become more and more driven by real-world experimental evaluations rather than network simulation. Numerous testbeds of WSNs have been set up in the past decade, often with very much differing architectural design and hardware.**

The Testbed Management Architecture for Wireless Sensor Networks (TARWIS) presented in this paper provides the most crucial management and scheduling functionalities for WSN testbeds, independent from the testbed architecture and the sensor node's operating systems. These functionalities are: a consistent notion of users and user groups, resource reservation features, support for reprogramming and reconfiguration of the nodes, provisions to debug and remotely reset sensor nodes in case of node failures, as well as a solution for collecting and storing experimental data. We describe the workflow of using a TARWIS on a WSN testbed over the entire experimentation life cycle, starting from resource reservation over experiment definition to the collection of real-world experimental data.

***Index Terms*—Wireless Sensor Networks, Testbed Experimentation, Testbed Federation, Network Management**

I. Introduction

In the wireless sensor and ad-hoc network community, network simulation tools have generally been identified and criticized for only providing a limited degree of realism. Researchers today generally aim at proofing the feasibility of their proposed protocols and mechanisms on real-world devices. For evaluations of protocol behavior in practice, real-world sensor network testbeds have become indispensable.

In the past decade, the most frequently used and cited experimental WSN testbed deployments were either Harvard University's MoteLab [1], the TWIST testbed [2] of TU Berlin or the Kansei [3] testbed of Ohio State University. The management software solutions implemented for these testbeds have, however, generally been tightly coupled to one particular testbed deployment, and are not easily reusable for further testbed setups. Still today, researchers setting up a testbed are often starting from scratch to implement testbed management features, which are as simple as user account management, experiment resource reservation, configuration and scheduling, or a consistent representation of results.

II. The Testbed Management Architecture for Wireless Sensor Networks (TARWIS)

We bridge this discovered gap with our TARWIS management architecture presented in this paper. TARWIS has been kept independent of the underlying testbed organization or the sensor node hardware and software. In [4], we thoroughly discuss the software architecture of TARWIS, and outline the advantages of the employed technologies over that of existing testbed management solutions. These main advantages can be summarized as follows:

- TARWIS has been kept as independent as possible of most crucial design questions of its underlying testbed hard- and software. TARWIS only requires *that* the nodes can be remotely controlled from within the portal server, it makes no restrictions *how* (i.e., using USB cables, Ethernet, 802.11 wifi, etc.) this is actually achieved.
- TARWIS allows the testbed user to monitor and interact with the ongoing experiment at *run-time*. Nodes can be *reprogrammed*, *reconfigured* or *hard-reset* remotely over the browser window during the entire experiment run-time.
- TARWIS offers an integrated and federal approach for user authentication, authorization and account management basing on Shibboleth [5]. Nine WSN testbeds of the WISEBED [6] project form the WISEBED testbed federation, with node deployments of several 10 to more than 100 nodes. Each user account of the Shibboleth federation can be used for all testbeds.
- TARWIS fully integrates the Wireless Sensor Network Markup Language (WiseML) [7], an XML standard schema for describing experimental data in WSNs.

The WISEBED project, as well as the integration of TARWIS into the WISEBED testbed software architecture is described in detail in [8].

III. Experimentation using TARWIS

This section illustrates the workflow of using TARWIS to schedule, configure and run an experiment on any TARWIS-administered testbed. The subsequent screenshots have been made at the TARWIS deployment of University of Bern.
The TARWIS Web Interface has four main tabs for actions related to *Reservation*, *Experiment Configuration*, *Experiment Monitoring*, and *Testbed Management*, as depicted in the upper half in Figure 1. Depending on the role of the user in the testbed (visitor, user, administrator), access rights are defined such that some tabs are accessible and some are not (e.g. the testbed management tab is only accessible for administrators).

A. Reservation

The *Reservation* tab offers a user interface for querying and manipulating the Web Services-based TARWIS Resource Reservation system. The main reservation overview screen is depicted in Figure 1. The screenshot per default depicts the current day. The screen lists the node resources, sorted by the node types, on the y-axis versus the time on the x-axis. The time is divided into indivisible units of 15 minutes.
At that particular day, the testbed of University of Bern consisted of 21 TmoteSky/TelosB [9]. In the meanwhile, the

978-1-4673-0269-2/12/$31.00 © 2012 IEEE

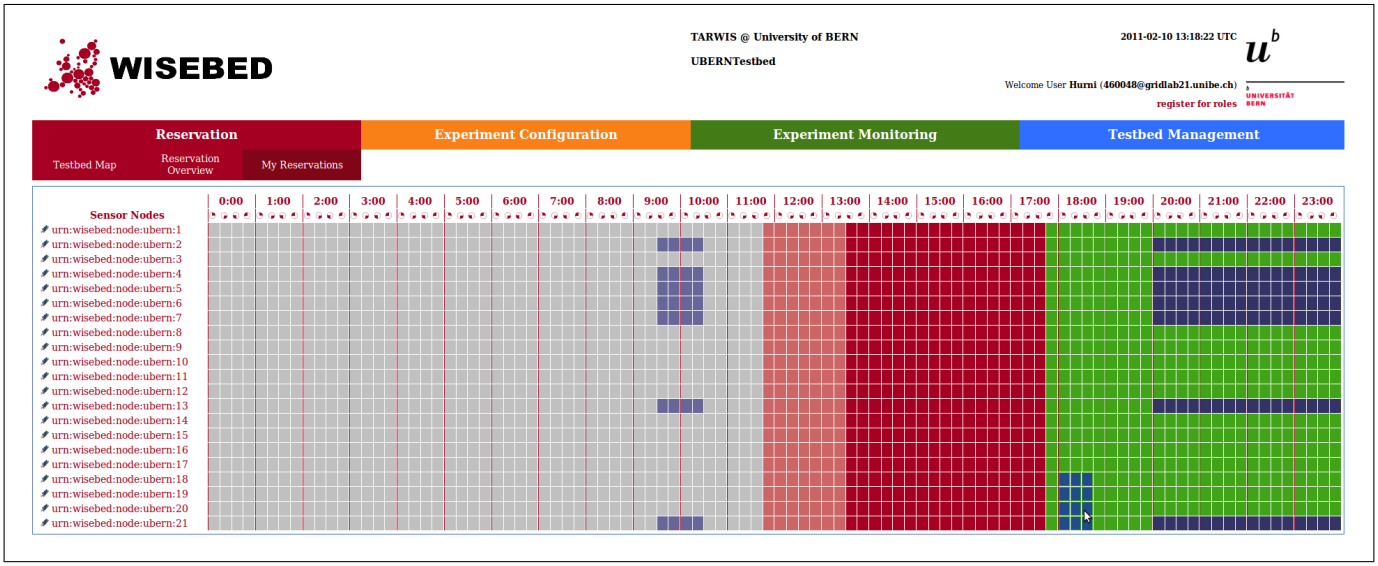

Fig. 1: TARWIS Reservation Screen

network has been extended to roughly 50 nodes, including 7 MSB430 [10] sensor nodes.

Figure 1 depicts three scheduled reservations, of which one is already past, one is ongoing. The experiments scheduled by the current user are colored in dark blue. The user taking the screenshot obviously had scheduled two reservations on a subset of 7 TmoteSky/TelosB nodes (listed on the left). The free time slots are colored green, whereas the time that is already past and that has not been allocated is colored grey. Past or partly-past reservations are colored with a grey shade, as clearly visible in Figure 1 before 12.30 UTC. At the time of accessing the testbed, an experiment of another testbed user that is using all nodes at the same time is currently running, which is indicated by the *red* reservation between 13.30 and 17.45 UTC.

Clicking on a rectangle and dragging with the mouse cursor selects a rectangle of nodes and time units, which define the time and the affiliated resources of that particular reservation. Nodes 18, 19, 20 and 21 are just about to be selected for a reservation starting at 18.00 UTC, as displayed in the bottom right of the figure.

B. *Experiment Configuration*

The third tab in the TARWIS Web Interface implements the *Experiment Configuration* process. All user-specific experiment and configuration data can be manipulated in this tab. In the first sub-tab, the user can store sensor node code images - the binary images, which are usually compiled with the mspgcc toolchain [11] for the majority of the sensor node platforms. These images are uploaded to the database with a user-supplied name and a unique identifier. The second sub-tab of the *Experiment Configuration* tab relates to the configuration (or modification) of the scheduled reservations of the visiting TARWIS user. Figure 2 depicts the screen where the user can define the experiment configuration for the reservation he/she scheduled beforehand. On the left, each sensor node has to be assigned a sensor node code image. One can assign one image to all nodes, or configure nodes

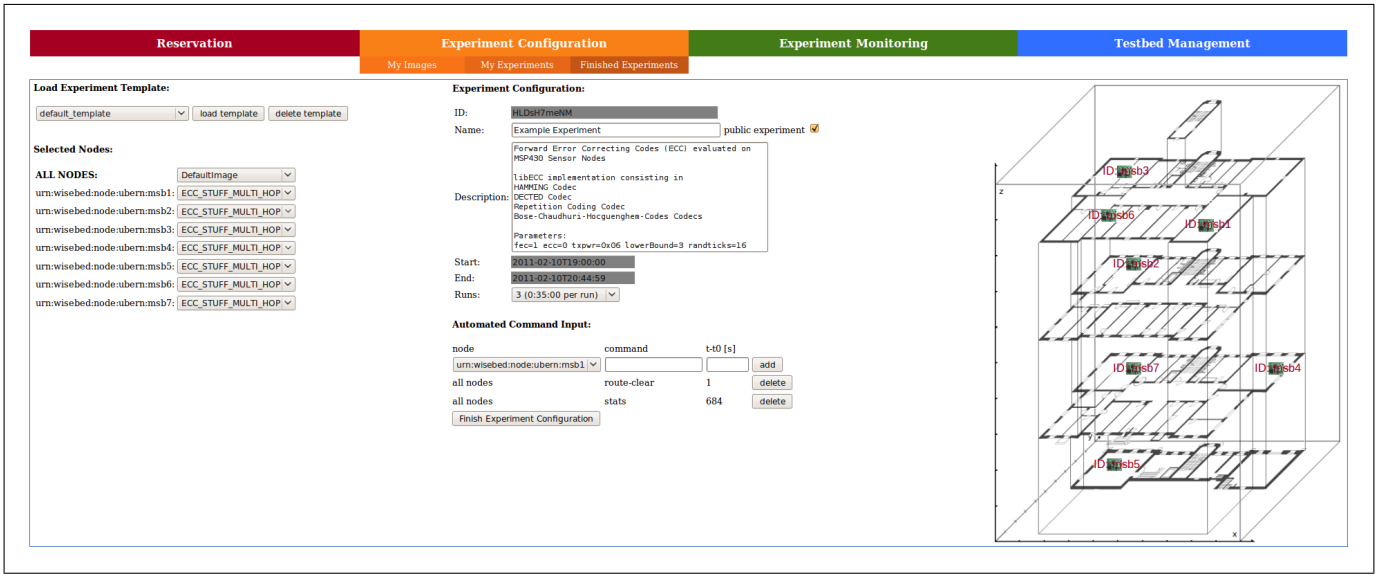

Fig. 2: TARWIS Experiment Configuration Screen

individually. In the middle of the screen, the experiment can be given a name and a brief description in the input and text fields. With the *public* checkbox, the user can let other users observe his experiment at run-time, and also permit other users to later download the experiment results. The map on the right depicts the selected nodes and their position within the testbed.

Experiment Runs: As researchers often have to carry out experiments several times in a row in order to obtain statistically significant results, TARWIS integrates the *Runs* option in order to easily define how many *experiment runs* the experiment shall consist of. The entire duration of the reservation is then split into this specified number of runs. After each run, the output is written to an output file and subsequently added to a zip archive. This archive is then made available for download and sent to the user as an attachment of an email.

Automated Commands: On the bottom of the page just below the experiment description box and the *Runs* option, the user may add so-called *automated commands*. These commands are issued after a specified number of seconds after the flashing operation finished. The target destination of these *automated commands* can be specified in the dropdown-box. A user may broadcast a command to all the sensor nodes in his selected set of nodes, or select only one unicast command recipient. The option is particularly useful in case an experiment needs to be started synchronously, e.g. by sending a *startExperiment* command to all nodes at the same time in order to initiate an algorithm.

Templates: Entering the configuration data for an experiment can become an exhaustive and time-consuming job, especially if there are many nodes that need to be reprogrammed with different code images, if all the input text fields have to be edited and if many automated commands need to be scheduled. Practice has shown that these steps can become tedious and repetitive especially in the case where many different but similar experiments need to be scheduled. In TARWIS, the user is hence given the opportunity to save the experiment settings as a *template*. When later scheduling another similar experiment, he/she may just select this saved template and reload it with the *load template* button. The previously saved experiment configuration is then loaded automatically (experiment description, run settings, public settings, automated commands, assignment of images to the nodes, etc.) and the user can just modify the changing parameters to reflect the new experiment settings.

My Experiments/Finished Experiments: When the user has finished configuring the experiments, he/she presses the *Finish Experiment Configuration* button. In the *My Experiments* subtab, he/she may still edit its pending experiment configurations given that the experiment has not yet started. The results of the finished experiments are available for download in the *Finished Experiments* tab.

C. Experiment Monitoring

The *Experiment Monitoring* tab offers to monitor running experiments at run-time. It is definitely one of the most crucial advantages of TARWIS over other testbed management systems, which usually run experiments in batch mode without giving the user the opportunity to monitor or even interact with them. Figure 3 depicts an excerpt of a running experiment in the *Experiment Monitoring* tab. With different experiments running on different subsets of the testbeds, one can choose the experiment to be observed, and switch between them arbitrarily. In the left part of the figure, an illustration of the testbed with the selected nodes positioned in the building of the testbed is displayed. On the top of the figure, the user can see the so-called *Experiment Controller Output* log. This text field logs the status messages of the TARWIS Server Daemon experiment-dedicated subprocess, which emits certain status

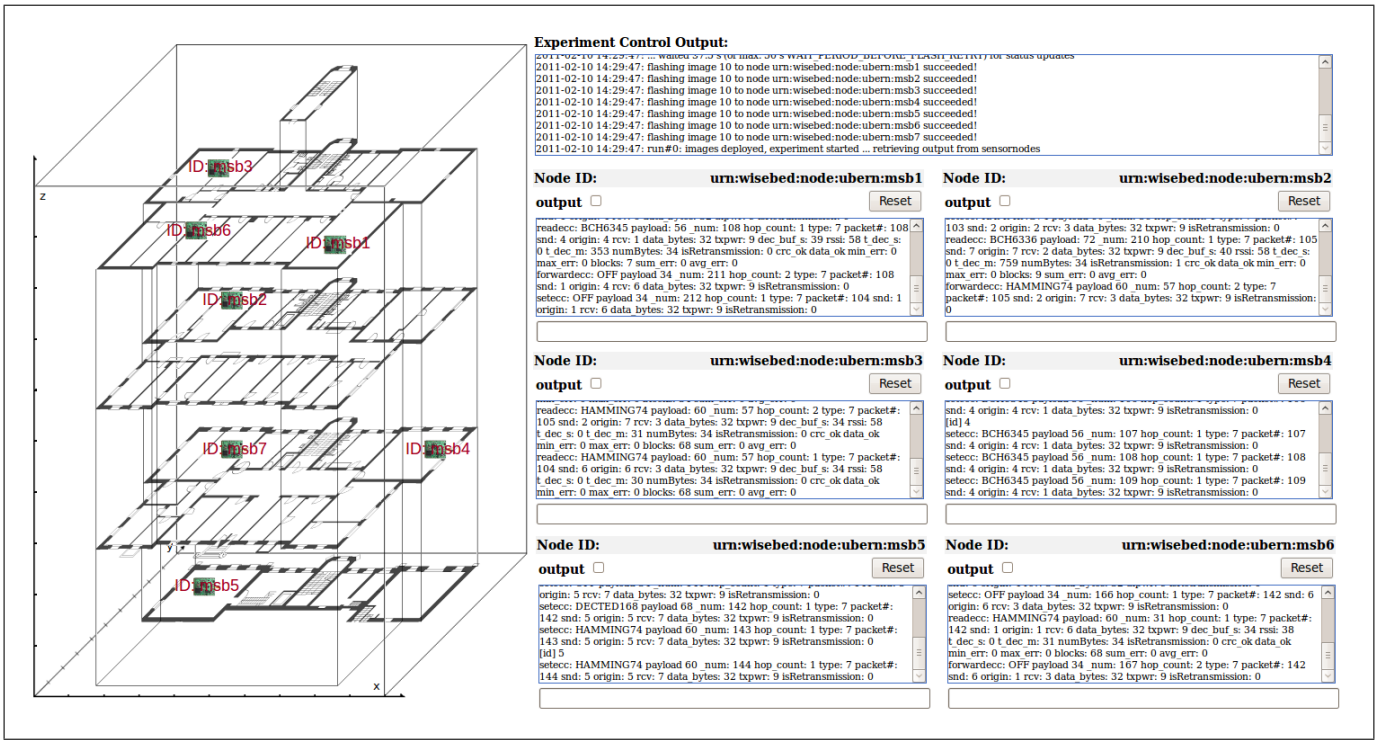

Fig. 3: TARWIS Experiment Monitoring Screen

information, e.g. at what time the nodes where reprogrammed (*flashed*), whether the operation was successful, which run is currently being executed, and much more. On the right of the figure, the user can monitor the output of the selected sensor nodes. The six displayed output windows list the last 20 lines of each node, which were captured from the sensor nodes' serial interfaces. This output information is particularly useful for debugging sensor node applications. Our practice has shown that with observing the output windows from the individual sensor nodes concurrently at run-time, erroneous and unintended behavior and possible reasons for the latter can be much easier identified, compared to tedious offline trace analysis in log files.

Below the output windows of the nodes, the user can enter commands to the sensor nodes, which are subsequently sent to the sensor nodes via the backend implementation and written to their serial interfaces. The user can hence interact with the nodes and, e.g., obtain status information using the TARWIS web interface in order to get a clearer picture why the examined protocol is not behaving as expected.

IV. Experiment Results in TARWIS

TARWIS retrieves the experiment output and subsequently stores this output in a database. When the experiment expires, all the retrieved output is exported to a file adhering to the Wireless Sensor Network Markup Language (WiseML). This WiseML file comprises all the significant information about an experiment, e.g., where the experiment took place geographically, what kind of nodes were used, what their configuration was, and much more. Using the Wireless Sensor Network Markup Language (WiseML) [7] standard for the representation of the results further allows for making the experiment data public to research partners in a common well-defined language, giving them the opportunity to repeat the same or similar experiment and e.g. trying to improve results. WiseML is a cornerstone towards a unified representation of experimental data in the WSN field, may it be from experiments on simulators or real-world data traces. Three network simulators have to-date adopted the WiseML standard, among them the popular Contiki simulator COOJA [12].

V. Conclusions

In this paper we have presented TARWIS, our system for testbed-based experimental research in WSNs, and we have thoroughly discussed its workflow for conducting experimental evaluations based on it's convenient and clear user-interface. Our practical experiences have shown that using TARWIS, researchers working in the field of WSNs have a powerful instrument for prototyping and evaluating various sensor network protocols and mechanisms. The option to monitor experiments and interact with the entirety of the sensor nodes at run-time using a web browser is a powerful and unique feature of TARWIS and has not existed in any of the reviewed testbed management solutions before. The availability of the system notably expedited experimentation with our real-world distributed WSN testbed facilities. The WISEBED testbeds are to date regularly used by students as well as international researchers. Research results from these testbeds have yet been used for numerous scientific publications, e.g. [13][14].

Acknowledgments

This work was supported by the European Commission under contract IST-2008-224460 and the Swiss National Science Foundation (SNF) under contract 200021-126718.

References

[1] G. Werner-Allen, P. Swieskowski, and M. Welsh, "MoteLab: a Wireless Sensor Network Testbed." ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), Los Angeles, USA, April 2005.

[2] V. Handziski, A. Koepke, A. Willig, and A. Wolisz, "TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Network." International Workshop on Multi-hop Ad Hoc Networks (REALMAN), Florence, Italy, May 2006.

[3] E. Ertin, A. Arora, R. Ramnath, and M. Nesterenko, "Kansei: A Testbed For Sensing At Scale." ACM/IEEE International Conference on Information Processing In Sensor Networks (IPSN), Nashville, USA, April 2006.

[4] P. Hurni, M. Anwander, G. Wagenknecht, T. Staub, and T. Braun, "TARWIS - A Testbed Management Architecture for Wireless Sensor Network Testbeds." International Conference on Network and Service Management (CNSM), Paris, France, 2011.

[5] Shibboleth: A Standards-based Open-Source Internet2 Middleware Architecture Project.

[6] Seventh Framework Programme FP7 - Information and Communication Technologies, "Wireless Sensor Networks Testbed Project (WISEBED)," FP7 Project 2008-2011. [Online]. Available: http://www.wisebed.eu

[7] Deliverable D4.1: First Set of well-designed Simulations, "Experiments and possible Benchmarks," June 2008. [Online]. Available: http://www.wisebed.eu

[8] G. Coulson, B. Porter, I. Chatzigiannakis, C. Koninis, S. Fischer, D. Pfisterer, D. Bimschas, T. Braun, P. Hurni, M. Anwander, G. Wagenknecht, S. P. Fekete, A. Kröller, and T. Baumgartner, "Flexible experimentation in wireless sensor networks," *Communications of the ACM*, vol. 55, Jan. January 2012.

[9] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling Ultra-Low Power Wireless Research." International Conference on Information Processing in Sensor Networks (IPSN), Los Angeles, USA, April 2005.

[10] M. Baar, E. Koeppe, A. Liers, and J. Schiller, "The ScatterWeb MSB-430 Platform for Wireless Sensor Networks." SICS Contiki Workshop, Kista, Sweden, 2007.

[11] mspgcc - A port of the GNU tools to the Texas Instruments MSP430 microcontrollers. [Online]. Available: http://mspgcc.sourceforge.net

[12] J. Eriksson, F. Osterlind, N. Finne, N. Tsiftes, A. Dunkels, and T. Voigt, "COOJA/MSPSim: Interoperability Testing for Wireless Sensor Networks." International Conference on Simulation Tools and Techniques (SimuTools), Rome, Italy, March 2009, pp. 27:1–27:7.

[13] P. Hurni, U. Bürgi, T. Braun, and M. Anwander, "Performance Optimizations for TCP in Wireless Sensor Networks." European Conference on Wireless Sensor Networks (EWSN), Trento, Italy, February 2012.

[14] P. Hurni and T. Braun, "On the Accuracy of Software-based Energy Estimation Techniques." European Conference on Wireless Sensor Networks (EWSN), Bonn, Germany, February 2011.