TARWIS - A Testbed Management Architecture for Wireless Sensor Network Testbeds

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*—With research on Wireless Sensor Networks (WSNs) becoming more and more mature in the past five years, researchers from universities all over the world have set up testbeds of wireless sensor networks, in most cases to test and evaluate the real-world behavior of developed WSN protocol mechanisms. Although these testbeds differ heavily in the employed sensor node types and the general architectural set up, they all have similar requirements with respect to management and scheduling functionalities: as every shared resource, a testbed requires a notion of users, 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. The TARWIS management architecture presented in this paper targets at providing these functionalities independent from node type and node operating system. TARWIS has been designed as a re-usable management solution for research and/or educational oriented research testbeds of wireless sensor networks, relieving researchers intending to deploy a testbed from the burden to implement their own scheduling and testbed management solutions from scratch.**

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

I. Introduction

For years, simulation has been the research tool of choice in the majority of studies on wireless ad-hoc and sensor networks. However, with a widening gap between simulation results and real-world prototype results, the appropriateness of simulation tools for simulating wireless phenomena has more and more been been questioned. Especially in the wireless sensor and ad-hoc network community, inappropriate parameter settings and unrealistic radio, traffic and/or mobility models have been identified and criticized as a general drawback of simulation studies (c.f. [1] [2]). With research in this field growing more mature, researchers have generally aimed at proofing the real-world feasibility of their proposed protocols and mechanisms on real-world devices. For the purpose of evaluating protocol behavior in practice, experimental sensor network testbeds have become indispensable.

In the past five years, numerous universities and research institutions have started to set up real-world sensor network testbeds. In most cases, these testbeds have been set up for research and teaching purposes, in order to enable testing and evaluation of real-world behavior of developed protocol mechanisms. An increasing number of stationary WSN testbeds have been put into operation, with different node hardware and very heavily differing architectural testbed design. These design aspects include, but are not limited to the following questions and trade-offs:

- Is the testbed an indoor or outdoor deployment? In indoor deployments, nodes are often deployed in offices or other *safe* locations. Node outages are generally less likely than in hostile outdoor environments.
- Does the testbed setup require intermediate (gateway) nodes? Small mesh routers are often utilized to remotely reset or reprogram the sensor nodes, e.g. over the USB port.
- Is there a high-bandwidth backup channel, e.g., Ethernet or IEEE 802.11? Reliable and high-bandwidth backup channels are frequently utilized to obtain status information from the sensor nodes during the experiment runtime, or to trigger node reset or reprogramming operations.
- Are there provisions for remote access, e.g, over a web interface? While some testbeds are only accessible locally at the site of deployment, more and more testbeds have been made available to other people intending to use the testbed for their research purposes.

The testbed management solutions implemented for many of today's testbeds have generally been tightly coupled to one particular testbed deployment, and are not easily re-usable for further testbed setups - a bridge we intend to gap with the TARWIS management architecture described in this paper. TARWIS [3] is a flexible and generic testbed management system for wireless sensor network testbeds - a re-usable management solution for research and/or educational oriented research testbeds of wireless sensor networks. TARWIS features essential services for testbed operation, such as multi-user access to testbed resources, online reservation, experiment configuration and experiment scheduling, automated data acquisition and logging as well as real-time experiment observation.

II. Related Work

While there are certainly many more testbeds which would deserve to be mentioned, we briefly discuss the most prominent sensor network testbed deployments and their characteristics in this section:

MoteLab [4] is a sensor network testbed on the campus of Harvard University. It currently features roughly 200 TelosB [5] sensor nodes. All sensor nodes are wired to programming boards allowing for direct reprogramming and communication.

The TWIST testbed [6] resides in a building in the campus of TU Berlin and spans across several floors. The total number of sensor nodes belonging to the testbed is approximately 200, featuring two hardware types. The testbed is organized hierarchically in 3 tiers, consisting of servers, super nodes and sensor nodes. The testbed is open for job submissions to registered users. The employed management software *TWISTv1* is available to the public. However, the software is generally quite tightly coupled to the Ethernet-based wired setup of the testbed and the 3-tiered-architecture.

Kansei [7] is a sensor network testbed located at Ohio State University, which targets at research in large sensor networks. Currently, about 200 sensor nodes are deployed, along with the same number of gateway stations attached to each one of the sensor nodes. The testbed features a web-based interface for registered researchers for submitting jobs to the testbed.

PowerBench [8] is a testbed infrastructure specifically designed for benchmarking power consumption. It includes hardware and software components for capturing the power traces the nodes in the testbed in parallel, which can be used for later offline processing and debugging purposes.

The TutorNet testbed [9] uses a 3-tier network topology with testbed servers, gateway stations, and sensor nodes, which feature USB connections to the gateway stations. Each such station together with a number of sensor nodes (at most 7) forms a cluster. Currently, there are more than 100 nodes and gateway nodes deployed.

Sensei-UU [10] is a relocatable testbed designed to enable users to repeat experiments with mobile, heterogeneous nodes in diverse environments, in contrast to using a static indoor testbed with predefined hardware and sensor equipment. The testbed has been set up in different locations in and around the University of Uppsala campus.

III. TARWIS Features

When studying the beforementioned work, it becomes obvious that there has already been a lot of work dedicated to the setup of WSN testbeds, as well as for tailormade software for network administration tasks. One may therefore be urged to ask the question where the benefit of the TARWIS Testbed Management Architecture is. In this section, we pinpoint the main advantages of TARWIS over their predecessors, before we continue to describe TARWIS and its features in a more detailed manner in the subsequent sections.

a) While most testbed management solutions have been implemented around a specific testbed architecture and node type, TARWIS has been kept as independent as possible of most crucial design questions of its underlying testbed hard- and software. TARWIS can hence be used to control and manage testbeds with a single server architecture (to which nodes are connected e.g. over USB cables), with a two- or three-tiered architecture where gateway meshnodes control the access to the individual meshnodes, or to testbeds where no wired backup channel is present at all. TARWIS only requires *that* the nodes can be remotely controlled from within the portal server, it makes no restrictions *how* this is actually achieved. The TARWIS components communicate over standardized and programming-language independent APIs, which keeps the re-usability and generality of TARWIS on a high level.

b) Most of the above listed testbed management solutions interact with specific features of the node and its operating system, and thereby hence lose their generality. TARWIS is totally independent from the node type and node operating system. So far it been deployed on nine different testbeds throughout Europe, using five different node types and operating systems. Besides our testbed at University of Bern, to date there are eight other TARWIS deployments, often with heterogeneous sensor node types and controller hardware operable at the universities of Lancaster, Lübeck, Braunschweig, Berlin (FU), Delft, Geneva, Patras and UPC Catalunya. These testbeds are part of the pan-European testbed federation of the WISEBED [11] project.

c) Unlike other testbed management solutions, where experiments are usually set up and run invisibly in batch-mode, TARWIS allows the testbed user to monitor and interact with the ongoing experiment at run-time by observing the output of the selected sensor nodes in a browser window. TARWIS offers the same technical capabilities to the experimenting testbed user as if the sensor nodes would be attached to its desktop computer: nodes can be *reprogrammed*, *reconfigured* or *hard-reset* over the browser window at experiment run-time.

d) TARWIS has been designed to integrate testbeds from different universities or research institutions into the Shibboleth Federation [12] of the WISEBED [11] project. TARWIS offers an integrated and federal approach for user authentication, authorization and account management, and relieves the testbed operators from designing user management solutions from scratch. Each users account of the Shibboleth federation can be used for all testbeds deploying TARWIS, and for institutions deciding to join the Shibboleth federation in the future, rendering a per-site registration obsolete.

e) TARWIS fully integrates the Wireless Sensor Network Markup Language (WiseML) [13], an XML standard schema for describing experimental data in WSNs. In an attempt to achieve compatibility of sensor network experimental data with several simulation tools, the WiseML standard has been adopted by the well-known COOJA [14] simulator in [15]. WiseML is a cornerstone towards a unified representation of experimental data in the field, may it be from experiments on simulators or real-world data traces. TARWIS is to date the first architecturally generic and fully WiseML-compatible testbed management system.

IV. TARWIS Architecture

The architecture of TARWIS is illustrated in Figure 1. The figure displays the portal server, on which the essential parts of TARWIS are hosted. Besides the TARWIS server component (lower left corner), the portal server hosts the TARWIS web interface (within an Apache web server), which is protected by the authentication and authorization system Shibboleth [12]. In this section, we briefly describe the different components visible in Figure 1 along with the applied technologies.

A. Portal Server

The portal server is any customary desktop PC with some minimum 2 GB of RAM and a broadband Internet connection. TARWIS comes with installation scripts to simplify the setup which have been tested and optimized for Debian Linux [16] *Lenny* (v.5.0.0).

B. TARWIS Web Interface

TARWIS offers an intuitive and easy-to-use web-based user interface. This interface is implemented as a dynamic web page using PHP, offering the user convenient access to the testbed via a web browser. A MySQL Database Management System (DBMS) is used to store personal configurations and experiment definition data. In order to gain access to the TARWIS web interface, a user has to log in using its Shibboleth credentials. Based on the user identification credentials, TARWIS offers fine-grained access to the testbed resources - either as visitor (only being able to observe public-declared experiments), as a normal testbed user (being allowed to submit experiment jobs) or as a testbed administrator (a root-like account which - besides submitting experiments - is also capable of adding additional nodes, schedule maintenance tasks, delete normal user's experiments, etc).

C. Shibboleth

The TARWIS architecture separates the concerns of authentication and authorization to the resources. Authentication of users logging to the portals is based on Shibboleth [12]. When signing in to any TARWIS deployment, a user has to enter its user name and password at the so-called home organization. Each users' home organization is responsible for authenticating its affiliated users. Shibboleth is a standards-based, open source authentication and authorization system used for web Single-Sign On (SSO) and user and account management across organizational boundaries. To date, it is widely adopted in European education and research networks (e.g. SWITCH [17], DFN [18], eduGAIN [19]). Since recently, the most widely used computer science libraries IEEE Xplore [20] and Elsevier ScienceDirect [21] supports Shibboleth in order to facilitate access from universities worldwide. Using Shibboleth, users only require one account with user/password credentials to log in to all present TARWIS deployments.

D. TARWIS Server Daemon

The TARWIS Server Daemon is the main process controlling the entire experiment execution logic. It encapsulates the database access and all the logic operations defined in the WSDL description files. The TARWIS Server Daemon is written entirely in Perl [22]. It forks a designated subprocess as soon as any new experiment is scheduled. This subprocess then fetches the experiment description from the database, and checks with the TARWIS Reservation System whether the submitted experiment has a valid reservation. It determines which nodes have to be reprogrammed with which binary code image, then connects to the sensor nodes via the WSDL interface definitions defined in Reservation System, SessionManager and WSNService API in order to reset and reprogram them. Over this API, the testbed user is later given the opportunity to access and program the sensor nodes over the TARWIS Web Interface at run-time. Then, the TARWIS Server Daemon and its subprocesses retrieve the run-time experiment data from the sensor nodes, which is subsequently stored in specific tables in the database.

E. Web Services-based APIs/Interfaces

The TARWIS components communicate with each other over the established Web Services standard [23], using clearly-defined API descriptions available in the machine-processable format *Web Services Description Language (WSDL)*. Web Services interface descriptions are programming-language and operating-system-independent. There are Web Services/SOAP bindings for any major programming language on all major operating system platforms.
Figure 1 displays three examples of such API functions which are invoked by TARWIS: reprogramming a node using the *flashPrograms*, resetting nodes using *resetNodes*, and retrieving output or status messages from a node with *receiveStatus*. The implementation of these services can reside on the same computer as the TARWIS components, or any other computer that is accessible with high bandwidth from the portal server.

F. TARWIS Resource Reservation System

The TARWIS Reservation System is used to prevent concurrent access to the resources on the testbed, hence to ascertain uninterruptible experimentation of multiple users on the testbed. Reservations can be made for the entire testbed, or for subsets of testbed resources, with a subset consisting in at least one node. The TARWIS Reservation System exposes its primitives to the TARWIS Web Interface, where it can be manipulated in a browser window. Apart from that, it can also be queried and manipulated in a machine-driven manner (e.g., using a script) via the *RSService* Web Service. A fully interoperable reservation client, which communicates over the same WSDL functions, is furthermore available as iPhone

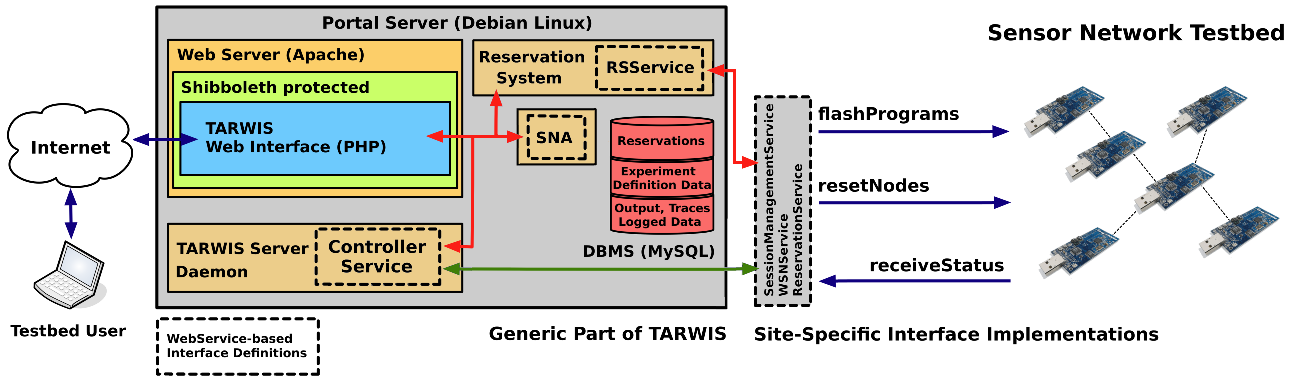

Fig. 1: TARWIS System Architecture - Testbed Generic Part (left) and Testbed Specific Implementation (right)

application, permitting to *book* time slots from within a simple smartphone. The ReservationSystem API consists in primitives to obtain a list of the current reservation (e.g. *getReservations*, *getConfidentialReservations*) and to schedule or manipulate reservations (e.g. *makeReservation*, *deleteReservation*).

G. TARWIS Sensor Network Authentication Service (SNA)

Each TARWIS deployment further exhibits a Web Service for authorization. The process of authorization - granting rights to authenticated users - is done locally for each TARWIS testbed. Each testbed operating organization (e.g., research institution or university) can specify which access right is granted for which specific user from any home organization in the WISEBED federation. To ensure modularity and consistency, the so-called Sensor Network Authorization Service (SNA) is hence present as a Web Service on the Portal Server. It can additionally be queried by any other Web Services enabled client, e.g. a Perl or Python script.

V. CONCLUSIONS

In this paper we have presented the TARWIS, the Testbed Management Architecture for Wireless Sensor Network Testbeds. TARWIS is a Web Services-based management system for administering and managing research testbeds of wireless sensor networks. TARWIS to date runs on six different testbeds of wireless sensor network testbeds of the WISEBED [11] project, with node deployments between a few 10 to more than 100 nodes, with a total number of sensor nodes of all TARWIS operated testbeds exceeding 1000. TARWIS is designed to federate testbeds of WSNs. The generic Web Interface, the standardized programming-language independent Web Service interfaces of the TARWIS backend system permit interested research groups to use TARWIS for their projected testbed, and to relieve them from the burden to implement own scheduling and testbed management solutions from scratch.

ACKNOWLEDGMENTS

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

REFERENCES

[1] S. Kurkowski, T. Camp, and M. Colagrosso, “MANET simulation studies: the incredibles,” *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 4, pp. 50–61, 2005.

[2] T. R. Andel and A. Yasinsac, “On the Credibility of Manet Simulations,” *IEEE Computer Magazine*, 2006.

[3] P. Hurni, G.Wagenknecht, M. Anwander, and T. Braun, “A Testbed Management System for Wireless Sensor Network Testbeds (TARWIS).” European Conference on Wireless Sensor Networks (EWSN), February 17-19, Coimbra, Portugal, 2010.

[4] G. Werner-Allen, P. Swieskowski, and M. Welsh, “MoteLab: a Wireless Sensor Network Testbed.” Information Processing in Sensor Networks (IPSN), 2005.

[5] J. Polastre, R. Szewczyk, and D. E. Culler, “Telos: Enabling Ultra-Low Power Wireless Research,” in *Intl. Symposium on Information Processing in Sensor Networks (IPSN)*, 2005.

[6] V.Handziski, A.Koepke, A.Willig, and A.Wolisz, “TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Network,” in *Intl. Workshop on Multi-hop Ad Hoc Networks (RealMAN 2006)*, Florence, Italy, 2006.

[7] E. Ertin, A. Arora, R. Ramnath, and M. Nesterenko, “Kansei: A Testbed For Sensing At Scale,” in *Intl. Conference On Information Processing In Sensor Networks (IPSN)*, 2006.

[8] Haratcherev, I., Halkes, G., Parker, T., Visser, O. and Langendoen, K., “PowerBench: a Scalable Testbed Infrastructure for Benchmarking Power Consumption.” International Workshop on Sensor Network Engineering (IWSNE), 2008.

[9] Tutornet, “A tiered wireless sensor network testbed,” http://enl.usc.edu/projects/tutornet.

[10] O. Rensfelt, F. Hermans, L.-A. Larzon, and P. Gunningberg, “Sensei-uu: a relocatable Sensor Network Testbed.” ACM Intl. Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization, 2010.

[11] Seventh Framework Programme FP7 - Information and Communication Technologies, “Wireless Sensor Networks Testbed Project (WISEBED),” Ongoing Project since June 2008, http://www.wisebed.eu.

[12] Shibboleth - an Internet2/MACE Middleware Project, http://shibboleth.internet2.edu.

[13] Deliverable D4.1: First set of well-designed simulations, “Experiments and possible Benchmarks. Technical report,” The WISEBED project group 2008, http://www.wisebed.eu.

[14] J. Eriksson, F. Osterlind, N. Finne, N. Tsiftes, A. Dunkels, and T. Voigt, “Cooja/mspsim: Interoperability testing for wireless sensor networks.” Intl. Conference on Simulation Tools and Techniques, 2009.

[15] Q. Li, F. Österlind, T. Voigt, S. Fischer, and D. Pfisterer, “Making wireless sensor network simulators cooperate,” in *7th ACM Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous networks (PE-WASUN)*, 2010.

[16] DEBIAN - The Universal Operating System, http://www.debian.org/releases/lenny/.

[17] The Swiss Education and Research Network (SWITCH), “Authentication and authorization infrastructure: System and interface specification,” 2004, http://www.switch.ch/aai/demo/.

[18] DFN, “Deutsches Forschungsnetz German Research Network,” http://www.dfn.de.

[19] eduGAIN, “GÉANT Authentication and Authorisation Infrastructure (eduGAIN),” http://www.edugain.org/.

[20] IEEE Xplore, “The IEEE Xplore Digital Library,” http://ieeexplore.ieee.org.

[21] Elsevier, “Elsevier ScienceDirect Library,” http://www.elsevier.com.

[22] The Perl Programming Language, http://www.perl.org/.

[23] The World Wide Web Consortium (W3C), “Web Services Activity Group,” http://www.w3.org/2002/ws/.