The tube.zero Deployment Support System

(Demo Abstract)

Andreas Reinhardt*, Parag S. Mogre†, Paul Baumann*, Johannes Schmitt*, Ralf Steinmetz*
*Multimedia Communications Lab, Technische Universität Darmstadt
Rundeturmstr. 10, 64283 Darmstadt, Germany
{andreas.reinhardt, paul.baumann, johannes.schmitt, ralf.steinmetz}@kom.tu-darmstadt.de
†Siemens AG, Corporate Technology T DE IT 2
Otto-Hahn-Ring 6, 81739 München, Germany
parag.mogre@ieee.org

***Abstract*—Practical experimentation in wireless sensor networks is generally challenging. The experimenter needs to install firmware images on each node individually, manually deploy and collect the devices, and retrieve debugging data logged on the nodes. To simplify experimentation, testbeds (fixed sensor network deployments) have become popular. Testbed management systems are a necessity to efficiently use the testbed resources. In this paper, we present the tube.zero deployment support system. Tube.zero supports the visualization of sensor node location and node connectivity. It also allows users to easily instantiate new experiments on any subset of nodes in the testbed and with the required network topology. Its user management permits temporal and spatial multiplexing of nodes for different jobs. On the user interface, experimenters can monitor the execution of their applications in real time and transmit control commands to each node. Tube.zero caters for the automatic distribution and installation of new firmware images as well as the retrieval of consolidated log files after experiments have completed.**

I. Introduction

As in-situ debugging capabilities of current wireless sensor nodes are typically confined to blinking LEDs, the real-world validation of simulation experiments is a non-trivial task. The fixed deployment of sensor networks in testbeds is thus a viable approach to maintain convenient access to the devices while exposing them to real-world characteristics. The deployed nodes are generally interfaced to a secondary network solely used for debugging purposes. This secondary channel can be realized over wired [1] or wireless channels [2].

To run an experiment on a testbed, the developer is typically confronted with the need for manual interaction, e.g. selecting nodes without previous knowledge of already running or scheduled jobs, dealing with the connectivity between the nodes, and handling the simultaneous distribution of firmware to the nodes. A reliable and intuitive deployment support system is hence necessary to increase the efficiency in resource sharing. Especially when many experimenters use the testbed, the need for coordinated temporal and spatial multiplexing of user experiments becomes obvious.

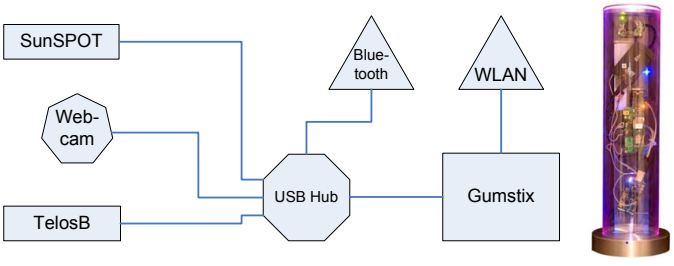


Fig. 1. Simplified structure and photo of a Tubicle node

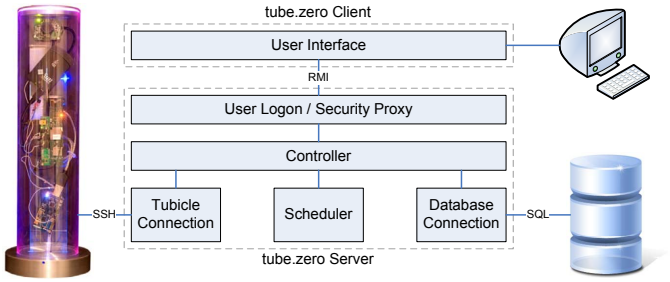


Fig. 2. Architecture of tube.zero

We have presented our TWiNS.KOM testbed in [3], which is currently composed of 20 Tubicle nodes (see Fig. 1 for a system overview). Tubicles provide three sensor platforms (Gumstix, SunSPOT, TelosB), which can be individually programmed. The presented tube.zero deployment support system has been developed to coordinate experiments on the testbed.

II. The tube.zero System

Tube.zero is based on the modular client/server architecture shown in Fig. 2. The tube.zero core is executed on a server, maintaining a consistent job schedule as well as the connections to all sensor nodes and the database. The client devices only execute the management interface and present the graphical user interface to the user. More details are provided at http://www.kom.tu-darmstadt.de/~reinhard/tubezero.

A. Node Status

All nodes are visualized on a graphical user interface, where three colored circles indicate the availability of each of the integrated platforms on the Tubicle, as shown in Fig. 3(a). While green signals an available device, a purple colored circle indicates the execution of a job on the platform. Nodes marked in red have not responded to connection attempts and are thus considered unavailable.

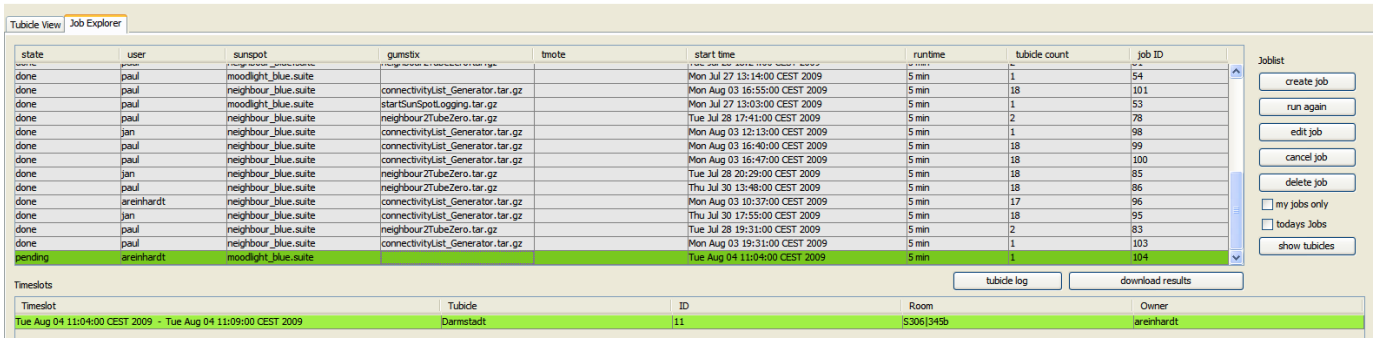


Fig. 4. The tube.zero user interface shows a listing of done, pending, and running jobs

B. Task Scheduling

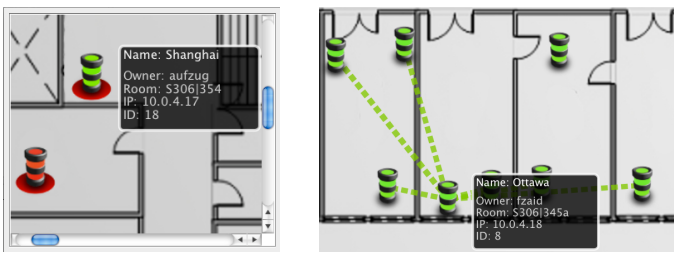
To create a new experiment, the user needs to select the number of nodes required and the experiment duration (ranging from five minutes to 24 hours). Based on this information, the task scheduler allocates the required number of nodes at the earliest time available, taking already scheduled jobs into consideration. An initial experiment topology based on node availabilities is suggested by the scheduler, however a manual reconfiguration to the user's preferred topology is possible. The user can optionally also shift the execution of the experiment to a later time, such as to conduct experiments during night time, where less cross traffic is present. Periodic jobs can also be created, and already finished jobs can be re-run with identical or modified properties easily. The scheduler adds all jobs into the job list shown in Fig. 4, which provides an overview of job details like owner and status.

The tube.zero system is provided with an option to upload a pre-compiled firmware image with connectivity assessment functionality on all nodes. Information on the node connectivity is then shown during job creation, depicted in Fig. 3(b). The integrated connectivity analysis is a helpful tool to avoid experimenting with disconnected networks, and provides the option to schedule this job periodically to update the informations in a given time interval.

C. Task Deployment

Before any new experiment can be deployed, the preceding job is stopped. All resulting log files are transferred to the server, where they are consolidated and stored in the database, kept available for later retrieval. Once the user has created a new task, the deployment process is triggered at the desired starting time. The new firmware files are then uploaded to the corresponding platforms of each selected sensor node (Gumstix, SunSPOT, TelosB, or any combination of these). On platforms without a dedicated job allocation, a radio logger application is being uploaded to avoid interference through unexpected packet transmissions. It needs to be remarked that all unselected nodes remain available for other experiments.

Firmware images are compressed and uploaded to the nodes in a bulk transmission, where they are then decompressed locally and installed on the attached platform. As TelosB components have no fixed hardware address for their radio transceivers, the radio address is changed in the firmware image according to the node identifier to avoid address collisions. The firmware upload process is logged to allow the developer to determine if any problems have occurred.



(a) Status View (b) Connectivity View

Fig. 3. Node status and connectivity view

D. Node Control and Logging

To exclude misconfigured or misbehaving nodes from running experiments, tube.zero offers dedicated functions to selectively disable nodes in running jobs. Similarly, experiments running erroneous firmware images can be cancelled before their scheduled termination through the user interface.

Besides direct interaction with the nodes, tube.zero offers two logging components. The process of deploying firmware images on the nodes is logged into a file to identify errors at this stage of the experiment. All data directly output by the nodes during the experiment can be followed in real time by opening a dedicated log connection through the user interface. Alternatively, all log files can be retrieved from the database after the experiment has terminated.

III. Demonstration Setup and Requirements

The tube.zero system has been designed to manage a testbed of Tubicle sensor nodes, and currently controls the TWiNS.KOM testbed with 20 Tubicles at the Multimedia Communications Lab (KOM) at TU Darmstadt. However, the number of nodes can easily be reduced without loss of functionality. To optimally demonstrate tube.zero's capabilities, a representative testbed of at least four Tubicle nodes will be set up at the conference venue. A laptop computer with attached monitor will be used to display the user interface and interact with the system.

During the demonstration, the following functionalities in tube.zero will be part of our demonstration:

- Creation of the building map and assignment of node locations: The tube.zero system is adapted to the individual environment by providing it with a map of the underlying building and placing the nodes on this map.
- Status information retrieval: After the map has been established, status and availability information about the Tubicles can be retrieved and visualized on the user interface.
- Job creation: By deploying firmware images to the selected platforms in the Tubicle nodes, new functionalities can be easily instantiated. The notion of a *job* comprises the required firmware images as well as the time period during which they are executed. The creation of a job also demonstrates the functionality of our scheduling component to avoid collisions.
- Log file retrieval: After successful execution of a job, the log files can be retrieved from individual Tubicles for analyses by the user.

In terms of the demonstrator's size, all equipment (i.e., a laptop computer that acts as the tube.zero server, a Wireless LAN access point, and a number of Tubicles) should fit well on a regular table of at least 1.6 meters width. The demonstration setup is mostly self-contained, i.e. apart from the need for at least one mains outlet to provide electricity to the server and the Tubicles, no further infrastructure is required. The demonstrator does not require significant adaptation to the surroundings at the conference, and can thus be installed and configured in less than an hour.

IV. Conclusion

We propose to demonstrate tube.zero, our deployment support system for Tubicle nodes. In contrast to existing systems, tube.zero allows multiple users to experiment on the same sensor network, sharing the network in both temporal and spatial dimension. Its task scheduler automatically allocates experiments the earliest available time slot. However, dedicated experimentation periods can also be defined by the user. Logging data can be accessed both in real time during the experiment, as well as downloading a complete archive containing all log data after the job has finished. A connection assessment functionality prevents users from unintentionally operating on a partitioned network. A live demonstration of our testbed management tool will be given, and the developers will be available at the conference for a dialogue on the possibilities of using tube.zero in other testbeds. In future we plan to extend tube.zero to support dynamic job rescheduling when other jobs are canceled, as well as preemption of jobs for superusers.

References

[1] G. Werner-Allen, P. Swieskowski, and M. Welsh, "MoteLab: A Wireless Sensor Network Testbed," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005, pp. 483–488.

[2] J. Beutel, M. Dyer, M. Hinz, L. Meier, and M. Ringwald, "Poster Abstract: Next-Generation Prototyping of Sensor Networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004, pp. 291–292.

[3] A. Reinhardt, M. Kropff, M. Hollick, and R. Steinmetz, "Designing a Sensor Network Testbed for Smart Heterogeneous Applications," in *Proceedings of the 3rd IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, 2008, pp. 715–722.