Testbeds - Setup and Interfaces

TEP:	130
Group:	Testbeds Working Group
Type:	Documentary
Status:	Draft
TinyOS-Version:	All
Author:	Jan Beutel
Draft-Created:	14-Jun-2007
Draft-Version:	1.2
Draft-Modified:	2007-06-28
Draft-Discuss:	TinyOS Testbed WG <tinyos-testbed-wg@eecs.harvard.edu»

Note

This document specifies a Best Current Practices for the TinyOS Community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

Abstract

This memo describes the structure and interfaces required for TinyOS compliant testbeds.

1. Introduction

The testing and validation of embedded code on real hardware is key to successful development of TinyOS applications. Although popular and powerful for high level analysis, current simulation methods lack in terms of level of detail and accuracy when used accross multiple system layers where abstractions and models used are inherently imperfect and impair results. Methods such as hardware emulation commonly used in embedded system development allow the execution of code on a hardware platform and therefore can guarantee timing but are very limited in terms of scalability and often confined to a lab usage only.

Sensor network testbeds try to overcome these deficiencies by allowing to execute software code under realistic operating conditions on real hardware embedded in a target environment. This approach allows to generate a level of detail especially in respect to the whole system (radio. processor, storage, sensors, interfaces) and the wireless environment (noise, fading, shading, errors, failures) while maintaining a certain degree of scalability. Remote programming as well as a feedback of status and debugging information from the nodes using testbed infrastructure alleviates the need for integrated services in sensor network applications. Additionally testbeds allow to operate a set of nodes in a controlled environment, i.e. using temperature variations or a controlled wireless environment.

A typical testbed is made up of a number of nodes (?do we state amounts here?) connected to a central resource for control and logging that are deployed in a physical space (testing area). Typically the central resource is a server with integrated datalogging capability. Often a web front end serves as a simple control and visualization resource. For the submission of testing jobs and the analysis of testing results external tools are attached to the central resource using a standardized interface. This document serves as a key specification document for such an interface and its intended usage.

MISSING: Difference of a testbed vs. a desktop test (e.g. single nodes with a programmer or a simple usb grid)

Examples of currently used sensor network testbeds are MoteLab [1] and the Deployment-Support Network [2].

2. Testbed Usage

A testbed can serve different purposes depending on the development status and requirements of a specific project. While this document does not target to restrict such usage it defines a set of mandatory modes of operation that a testbed must be able to support:

Modes of Operation:

- Single Shot Operation

Execute a testing job consisting of an uploading of a specific code image onto a set of nodes, remote programming of nodes using a testbed resource, the synchronized start of this software, capture of resulting target response, the centralized storage and timestamping of all actions and target response, ending of test execution, notification of a user of the end of test execution, output of all stored data on demand.

- Repetitive (e.g. using continuous integration or for regression testing)

A concatenation of single shot testing jobs, that in practice often will be used with the variation of one or more parameters.

- Long Term Operation (Profiling)

Other Topics:

- Federated Testbeds (in multiple environments)
- Access/Resource Arbitration
- Scheduling of testing jobs

3. Testbed Services

Required Services:

- identification of target devices (presence, type, hw rev)
- programming of target devices

- resetting of target devices
- logging of target response (UART mandatory, IRQ optional)
- versioning/identification of uploaded software
- identification/versioning/archiving of testing jobs
- testbed status information
- identification of testbed services
- user authentification

Optional: - power measurements - stimuli - distributed scheduling of actions (at nodes)

4. Implementation

- Server, DB/Storage, Interface XMLRPC
- Connection fabric
- On- and offline logging
- Supplied Power
- Mote Hardware

THINGS TO DISCUSS

- ?Do we state minimum requirements?
- number of nodes
- power supply (fixed/batt)
- power profiling
- power on/off of targets? is simple reset/erasing enough?
- feedback channel capabilities (delay, errors, lost packets...)
- target control? is control of (writing to) targets required or is it an optional feature?
- scheduling of actions (time synched???)

5. Reference

6. Acknowledgments

7. Author's Address

Jan Beutel
Gloriastr 35
ETH Zurich
8092 Zurich
Switzerland

phone - +41 44 632 7032

email - j.beutel@ieee.org

8. Citations

Appendix A. Example Appendix

This is an example appendix. Appendices begin with the letter A.

[1] G. Werner-Allen, P. Swieskowski, and M. Welsh. MoteLab: A wireless sensor network testbed. In Proc. 4th Int'l Conf. Information Processing Sensor Networks (IPSN '05), pages 483-488. IEEE, Piscataway, NJ, April 2005.

[2] M. Dyer, J. Beutel, L. Thiele, T. Kalt, P. Oehen, K. Martin, and P. Blum. Deployment support network - a toolkit for the development of WSNs. In Proc. 4th European Workshop on Sensor Networks (EWSN 2007), volume 4373 of Lecture Notes in Computer Science, pages 195-211. Springer, Berlin, January 2007.