Regression Testing Framework for Wireless Sensor Networks

Wolf-Bastian Pöttner, Daniel Willmann, Felix Büsching, and Lars Wolf
Technische Universität Braunschweig | Institute of Operating Systems and Computer Networks
[poettner|dwill|buesching|wolf]@ibr.cs.tu-bs.de

Motivation

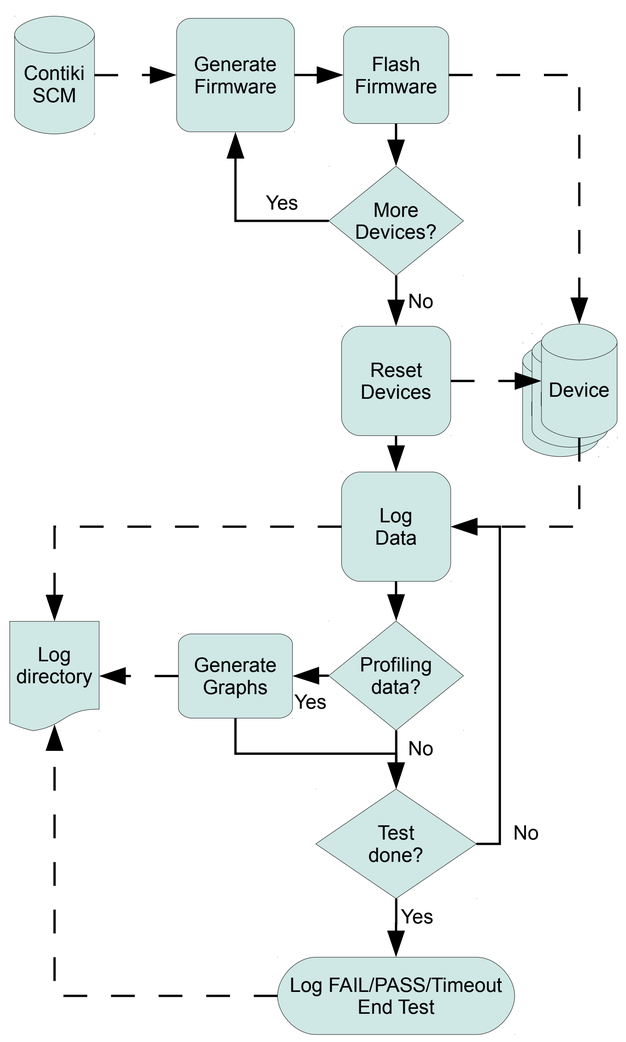
Program code for WSN nodes is becoming **more and more complex** and **quality assurance mechanisms** have to be installed to ensure a high code quality. Many WSN deployments do not have means to update code in the field. **Reprogramming of nodes** is often connected to significant effort and potentially **long timespans of unavailability**. **Continuous testing** of WSN software during development is indispensable to **ensure a high code quality**.

Simulators reproduce the real world with limited accuracy and therefore are only one tool for testing, debugging and performance evaluation. Especially generating **accurate information about execution time** of programs is **problematic on simulated nodes**. In our SenseApp paper we have shown, that **call graphs** for code running on actual nodes can be generated to **allow understanding of where a program spends most of its time**.

Approach

- Automatically trigger tests based on Source Code Management changes using Jenkins
- Tests are orchestrated by **the PC-based Test Framework**
- On the **nodes, test software** is used to perform the test and report results
- Enables automatic performance testing and running instrumented code for call graphs

Flowchart of a single Test Case (Simplified)



PC-based Test Framework

- **Compiles, flashes and executes** node-based test code
- Supports different source code files for different nodes
- **Simultaneous reset** of all nodes after flashing
- Ensure **defined initial state** for a test by flashing dummy image onto unused nodes for **reproducible test results**
- **Watch** for test results, timeouts, node reboots, performance numbers and profiling information
- **Collect and archive results** (logfiles, performance)
- **Modular design**, currently supports INGA and T-Mote sky

Node-based Test Code

- **Test application** written in native language for **maximum flexibility**
- Decides **on test success or failure**, reports via serial
- **Measures performance** and reports via serial
- Regular **log lines** for debugging
- Distinct **boot string** to detect reboots
- Based on but not limited to **Contiki OS**
- Can run tests **instrumented or non-instrumented**, call graph images are created by PC-based test framework

Jenkins CI

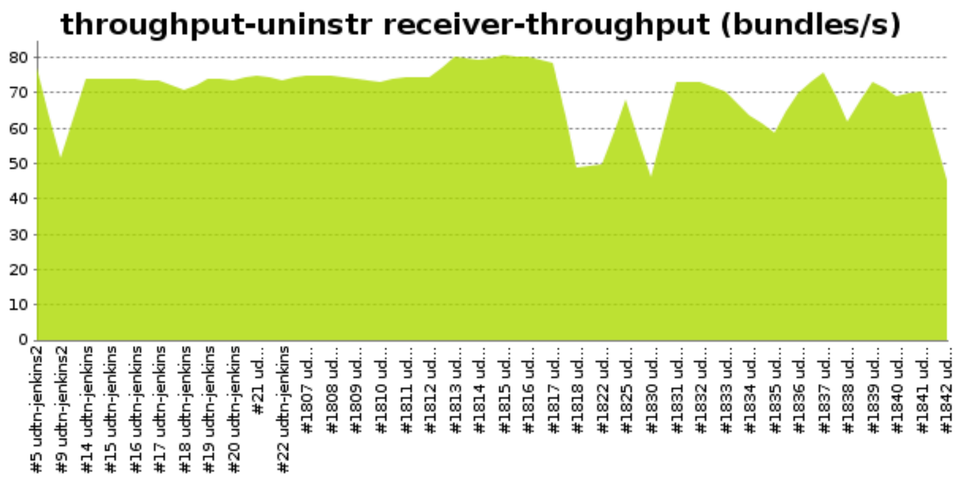
Jenkins **periodically polls the SCM** and **executes predefined tests** on changes. Additional jobs can be **executed regularly**. Workload is **automatically distributed onto multiple "slaves"** to allow running tests in parallel.

Jenkins **executes the PC-based test framework**, **collects the results** and **notifies the author** of commits if tests fail. Jenkins keeps track of performance figures and draws charts to visualize trends.

Jenkins

Logo: CC-BY http://jenkins-ci.org

Automated Performance Tests



Technische Universität Braunschweig

Institute of Operating Systems and Computer Networks