Demo Abstract: ProFuN TG: A Tool Using Abstract Task Graphs to Facilitate the Development, Deployment and Maintenance of Wireless Sensor Network Applications

Atis Elsts, Farshid Hassani Bijarbooneh, Martin Jacobsson, and Konstantinos Sagonas
Uppsala University, Sweden

***Abstract*—In this demo abstract we present ProFuN TG (Task Graph), a tool for sensor network application development using the data-flow programming paradigm. The tool has support for the whole lifecycle of WSN application: from the initial design of its task graph, task placement on network nodes, execution in a simulated environment, deployment on real hardware, to its automated maintenance through task remapping. ProFuN TG allows to program applications that incorporate quality-of-service requirements, expressed through constraints on task-to-task data flows.**

I. Introduction

Programming wireless sensor network applications is difficult, especially if certain reliability and data quality properties are desired together with energy efficiency.

We take an existing WSN programming methodology, the Abstract Task Graph [1], and implement it in ProFuN Task Graph [1] , a tool that facilitates the development of such applications. ProFuN TG not only allows the user to describe the functionality of an application with a task graph, but also to incorporate non-functional requirements [2] in that description. The requirements are expressed in form of probabilistic *constraints* on minimal packet delivery rate (PDR) and delivery delay, and set on data flows between tasks. The tool allows users both to use predefined tasks from a palette and to write their own tasks in C or SEAL programming languages.

The tool includes support for mapping these task graphs on network nodes, for macrocompilation of their code, and for their deployment both in simulated and real networks. The supporting run-time middleware gathers application performance statistics and determines whether the conditions of the constraints hold, enabling maintenance alert notifications, as well as automated maintenance through task remapping.

II. Architecture

Under the hood, ProFuN TG uses a number of well-known software tools and libraries: Contiki for system-level functionality, Cooja for network simulation, Gecode for constraint solving (used in the task allocation algorithm). For the visual frontend, an adapted version of Node-RED is employed.

ProFuN TG joins these components in a distributed microservice architecture (Fig. 1). The components communicate

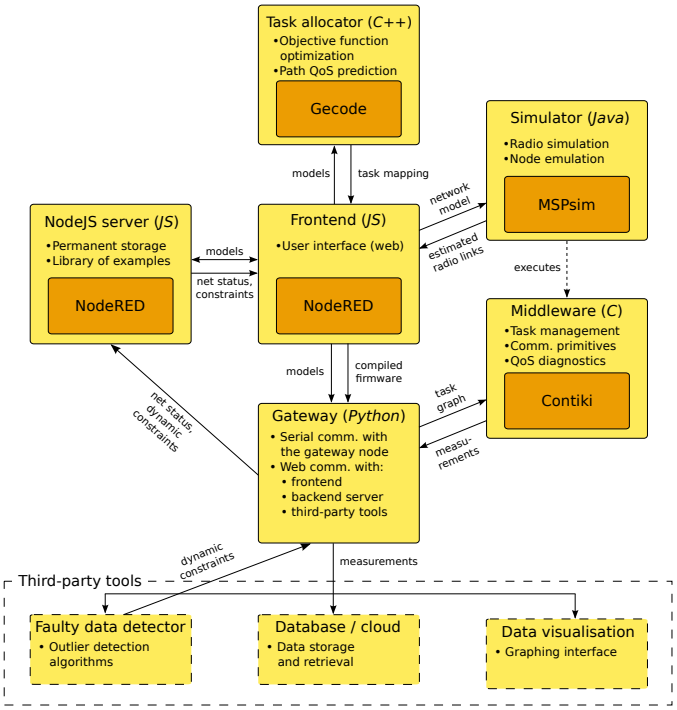


Fig. 1: Architectural overview

by passing JSON messages through HTTP, with the exception of WSN middleware, which uses an efficient binary format.

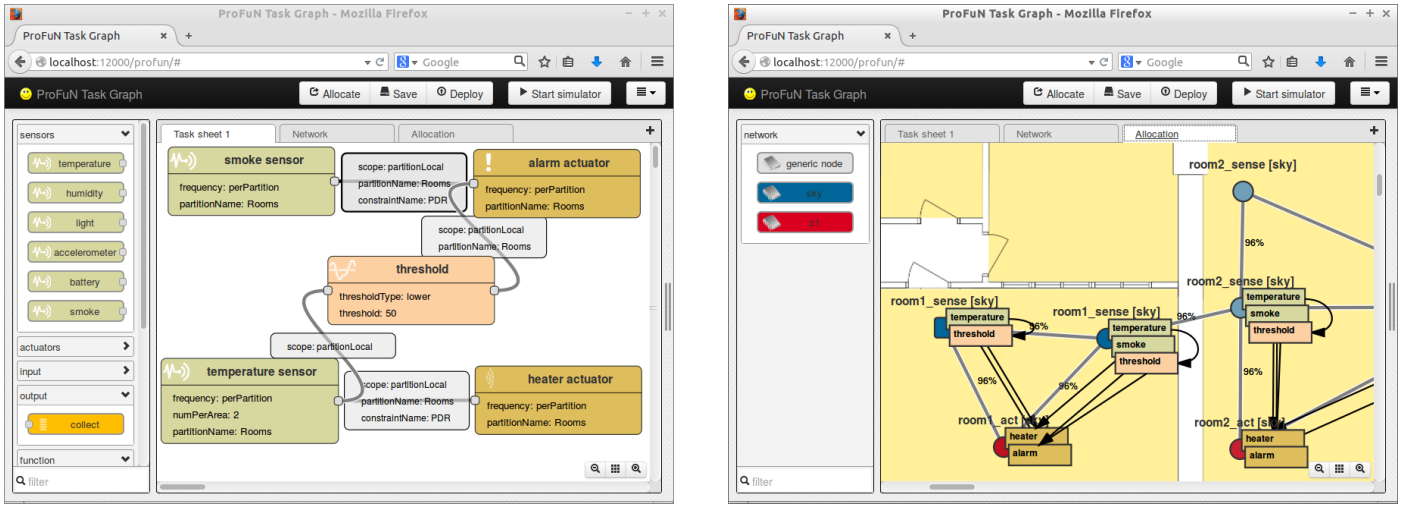
The tool provides an HTTP interface for data export in JSON format. Through it, custom or third-party tools can access the data, provide feedback for ProFuN TG , and impose dynamic constraints on the task mapping algorithm. In our demo setup we are going to use a custom principal component analysis (PCA) tool that detects sensor faults by learning acceptable sensor covariance bounds from past datasets.

ProFuN TG task sheet view (Fig. 2a) shows the task graph of a sample application. Network view (Fig. 2b) shows node placement in the network and tasks mapped on the nodes.

III. Application example

To explain the typical user workflow with the tool, let us take an example application that uses temperature sensors and

[1] http://parapluu.github.io/profun/



(a) Task graph view. Shows sensor, actuator, and processing tasks connected with data flows. On some of flows, PDR constraints are configured

(b) Network view. Shows sensing and actuating nodes (as blue and red circles, respectively) connected with network links (grey lines), and mapped tasks (rectangles) connected with data flows (black arrows)

Fig. 2: The visual interface of ProFuN TG showing a heater control application with fire detection

heater actuators to a maintain stable temperature in a number of rooms. The tool allows the user to:

- Create a task graph model for the application. The model consists of two tasks connected with a data flow.
- Set a constraint for minimally required PDR on the flow.
- Describe the model of the network: node locations, capabilities, and links between nodes. Probabilistic parameters such as link delay are described by probability distributions. In absence of explicit configuration, link existence and quality parameters are estimated by the simulator.
- Partition the network into regions (rooms) and configure task allocation frequency: one-pair-per-room.

Subsequently, the tool:

- Maps the tasks on network nodes, taking into account the network model, with the objective to minimize energy usage and satisfy the PDR constraints.
- If desired, simulates the setup to see if the constraints hold in the simulation environment.
- Deploys the task graph on a real WSN.
- Continuously tests for satisfaction of the constraints and requests task remapping when the test fails.

IV. Demo Setup

We plan to demonstrate a fault-tolerant light-sensing application developed with the tool.

The setup is going to consist of a laptop and a number of sensor nodes equipped with light sensors. On the laptop, ProFuN TG will be running, and the measured light intensity will be displayed. Light sensing and data collection tasks will be activated on the nodes.

Interested attendees are going to be invited to try to "break" the application by covering some of the light sensors and turning off some of the nodes, while the system is expected to demonstrate robustness by ignoring readings of the affected sensors and reallocating tasks to a different set of nodes.

V. Concluding Remarks

ProFuN TG enables design of data quality requirement-aware task graph applications by allowing the user to write PDR and delay constraints on data flows between tasks. The tool also enables deployment and maintenance of these applications by providing middleware that checks for faults at runtime and triggers reallocation in case a violation is detected.

A major difficulty for the tool to provide quality-of-service guarantees is caused by the fact that the inherent unreliability of wireless communications makes it hard to predict the performance of an application before actually deploying it. To make the runtime system more predictable, advanced link-layer protocols such as Glossy [3] should be used instead of the current Contiki network stack.

Acknowledgments

The authors acknowledge support from SSF, the Swedish Foundation for Strategic Research.

References

[1] A. Bakshi, V. K. Prasanna, J. Reich, and D. Larner, "The abstract task graph: a methodology for architecture-independent programming of networked sensor systems," in *Proceedings of the 2005 workshop on End-to-end, sense-and-respond systems, applications and services*. USENIX Association, 2005, pp. 19–24.

[2] F. H. Bijarbooneh, A. Pathak, J. Pearson, V. Issarny, and B. Jonsson, "A constraint programming approach for managing end-to-end requirements in sensor network macroprogramming." in *SENSORNETS*, 2014, pp. 28–40.

[3] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with Glossy," in *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*. IEEE, 2011, pp. 73–84.