Srijan: a graphical toolkit for sensor network macroprogramming

Animesh Pathak, Mahanth K. Gowda

Srijan: A Graphical Toolkit for Sensor Network Macroprogramming*

Animesh Pathak*
Project-team ARLES, INRIA Paris-Rocquencourt
Rocquencourt, 78153 France
animesh.pathak@inria.fr

Mahanth K. Gowda†
Institute of Technology, Banaras Hindu University
Varanasi, India 221005
mahanth.gowda.cse06@itbhu.ac.in

ABSTRACT

Macroprogramming is an application development technique for wireless sensor networks (WSNs) where the developer specifies the behavior of the *system*, as opposed to that of the *constituent nodes*. In this proposed demonstration, we would like to present *Srijan*, a toolkit that enables application development for WSNs in a graphical manner using data-driven macroprogramming. It can be used in various stages of application development, *viz.* i) specification of application as a task graph, ii) customization of the auto-generated source files with domain-specific imperative code, iii) specification of the target system structure, iv) compilation of the macroprogram into individual customized runtimes for each constituent node of the target system, and finally v) deployment of the auto generated node-level code in an over-the-air manner to the nodes in the target system. The current implementation of *Srijan* targets both the Sun SPOT sensor nodes and larger nodes with J2SE. Our demonstrattion will encourage users to perform end-to-end WSN application development on the SPOTs using *Srijan*.

Categories and Subject Descriptors: B.2.4 [Computer Systems Organization]: Computer-Communication Networks — Distributed Systems; D.3.3 [Software]: Programming Languages — Language Constructs and Features

General Terms: Design, Human Factors, Languages

Keywords: Sensor Networks, Toolkit, Macroprogramming

1. INTRODUCTION

Sensor network macroprogramming aims to aid the wide adoption of networked sensing by providing the domain expert the ability to specify their applications at a high level of abstraction. This is in contrast to the initial days of wireless sensor networks (WSNs), when application developers had to manually customize a set of node-level protocols to achieve system-wide goals, thus making the process difficult for the *domain experts*, who were not well-versed in the intricacies of distributed computing.

Since the goal of WSN macroprogramming research is to make application development easier for the *domain expert*, we believe that it is absolutely necessary to make *easy-to-use toolkits* for macroprogramming available to them in order to both make their task easier, as well as to gain feedback about the macroprogramming paradigms themselves. Although various efforts exist in literature for making WSN application development easier, very few general purpose graphical toolkits are publicly available for the application developer to choose from. We believe that toolkits supporting alternative paradigms will greatly aid the application developers, who will have a wide-range of programming styles to choose from, depending on application, as well as personal stylistic choice.

Our contribution, which we present in this demo, is *Srijan* – a graphical toolkit for WSN application development (named after the Sanskrit word for creation). Using it, the developer can create sense-and-respond applications for heterogenous systems using the data-driven ATaG [1] macroprogramming language.

2. BACKGROUND

The Abstract Task Graph[1] (ATaG) macroprogramming framework consists of an extensible, high-level programming model, a corresponding node-level run-time system, and a dedicated compilation framework to generate node-level code. An ATaG program is written in a *data-driven* manner using a mixed *imperative-declarative* programming model. The declarative portion of an ATaG program – a task graph – consists of the following components (see Figure 1 for details).

- Abstract Data Items: The main currency of information in an ATaG program. They represent the information in its various stages of processing inside a WSN.
- Abstract Tasks: These represent the processing per-

*This work was supported in part by the EC FP7 CONNECT project
*The author wishes to thank Viktor K. Prasanna, Amol Bakshi, and Qunzhi Zhou at the University of Southern California, Luca Mottola at the Swedish Institute of Computer Science, and Gian Pietro Picco at the University of Trento for their contributions to the toolkit.
†The work was performed during the author's stay at INRIA.

ESEC-FSE'09, August 23–28, 2009, Amsterdam, The Netherlands.

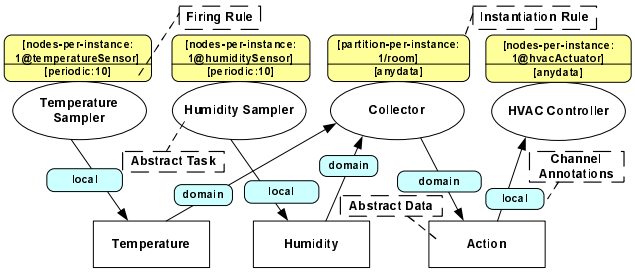

Figure 1: An ATaG program for building environment management.

formed on the abstract data items in the system. Tasks do not share state with other tasks, and can communicate only by producing and consuming data items. Tasks are annotated with *instantiation rules*, specifying where they can be located, as well as *firing rules*, specifying whether a task is triggered periodically or due to the production of certain data item(s).

- Abstract Channels: These connect tasks to the data items consumed or produced by them, and are annotated with logical scopes [3], which express the *interest* of a task in a data item.

The above task graph is complemented by *imperative code* for each data item and task. The developer uses this code to specify the processing that occurs when a task fires. Note that due to the data-driven programming model provided by ATaG, this imperative code does not have any inter-task communication function calls other than consuming or producing data items (using the `handleDataItemProduced()` and `putData()` primitives respectively).

The *input* to the *ATaG Compiler* [4] consists of the ATaG task graph, and the imperative code for each task and data item. In addition, the details of the target system, including the node locations and list of attached sensors and actuators, is also provided. The compiler then decides the placement of the tasks of the individual nodes. The *output* of this process is deployable code for each node, consisting of the tasks assigned to it. Additionally, the compiler generates *customized DART modules* for each node, containing the logical scopes where the data produced at the node is to be sent.

3. APP. DEVELOPMENT IN SRIJAN

We have developed the *Srijan* [5] graphical toolkit for WSN macroprogramming to allow application developers to i) easily specify their application in the form of a data-driven task graph in the ATaG macroprogramming language, ii) upload the details of their target network, and then iii) compile this high-level program to node-level code and iv) deploy it to the individual constituent nodes of their networked sensing system. The task graph for the application can be specified using our customization of the GME [2] generic modeling environment, and the imperative code for each task and data item is provided in Java. The toolkit produces code that is ready to be deployed on the Sun SPOT [7] nodes that run J2ME on the Sun Squawk virtual machine, in addition to larger nodes running J2SE JVMs. Figure 2 and Figure 3

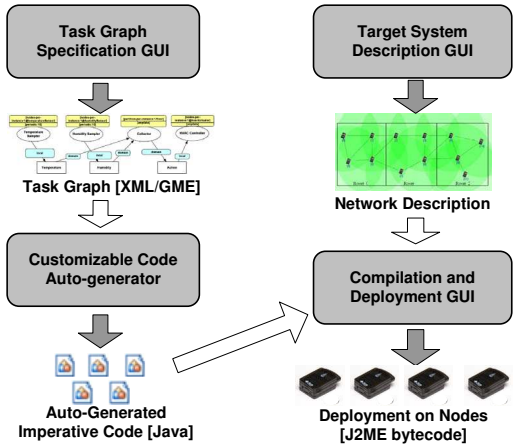

Figure 2: Overview of Application Development using *Srijan*

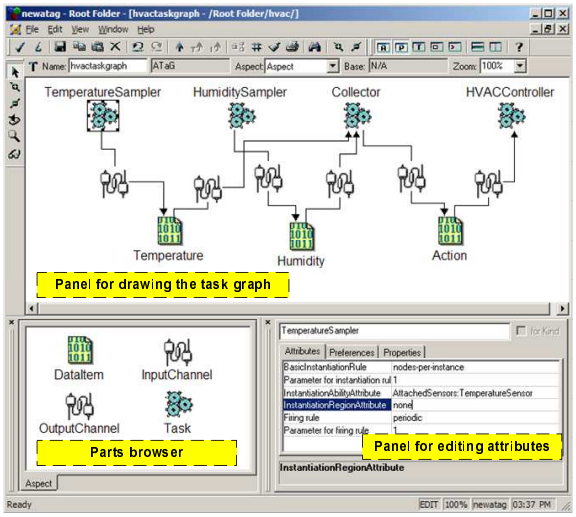

Figure 3: Task-Graph Description in Srijan.

depict some details of the steps involved in application development using the components of *Srijan*. Further details, including videos are available at the project website [6].

We would like to utilize the opportunity provided by this demo to get first-hand feedback on *Srijan* from users and application developers of the networked sensing domain, especially on what ease-of-use issues they face, and what features they would like. Additionally, we would be interested in the types of applications they would like to develop, which will guide us in our work on augmenting the ATaG language and its compiler.

4. 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 *Workshop on End-to-end Sense-and-respond Systems (EESR)*, 2005.

[2] The Generic Modeling Environment, http://www.isis.vanderbilt.edu/projects/gme.

[3] L. Mottola, A. Pathak, A. Bakshi, V. K. Prasanna, and G. P.

Picco. Enabling Scoping in Sensor Network Macroprogramming. In *4th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2007.

[4] A. Pathak, L. Mottola, A. Bakshi, G. P. Picco, and V. K. Prasanna. A compilation framework for macroprogramming networked sensors. In *Third International Conference on Distributed Computing on Sensor Systems (DCOSS)*, 2007.

[5] A. Pathak, Q. Zhou, and V. K. Prasanna. *Srijan*: A graphical toolkit for wsn application development. In *ProSenSe Workshop in the 4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), Santorini, Greece*, June 2008.

[6] *Srijan* - graphical WSN application development toolkit. `http://wsnsrijan.googlepages.com/`.

[7] Sun Small Portable Object Technology (Sun SPOT). `http://www.sunspotworld.com`.