End-to-end Toolkit for Developing a Class of WSN Applications on Sun SPOT Nodes*

Animesh Pathak†, Qunzhi Zhou†, Luca Mottola‡,
Amol Bakshi†, Viktor K. Prasanna†, and Gian Pietro Picco#

†University of Southern California, USA, {animesh, qunzhizh, amol, prasanna}@usc.edu

‡Politecnico di Milano, Italy, mottola@elet.polimi.it

#University of Trento, Italy, picco@dit.unitn.it

Abstract

Over the past years of research in Wireless Sensor Networks (WSNs), both the hardware used to construct WSNs and the languages used to describe their functionality have evolved. The Sun Small Programmable Object Technology (Sun SPOT) nodes are the latest offering in the former domain, with a Java virtual machine running on the metal. On the programming side, macroprogramming frameworks such as the Abstract Task Graph (ATaG) have been developed that aim to greatly reduce the burden of the application developer for a wide range of WSN applications.

In this work, we present an end-to-end solution *for macroprogramming WSN applications on Sun SPOT nodes using ATaG - a data-driven programming paradigm. We will demonstrate all the stages starting from the specification of the application to its compilation and deployment on actual sensor nodes, thus showcasing the power of our toolchain. We believe that our research will enable wide adoption of WSNs among a range of end-users who will now have a concrete end-to-end toolkit to develop WSN applications.*

1 Introduction

The Sun Small Programmable Object Technology [8] nodes (Sun SPOTs) are the latest hardware offering for Wireless Sensor Networks (WSNs), with a Java virtual running on the metal, and a small form factor. In the *macroprogramming* of WSNs, the application developer specifies the behavior of the resultant *system*, not the individual *nodes*. The various macroprogramming paradigms currently being developed include imperative programming [4], functional programming [5], task graphs [2], and spreadsheet-based programming [9], among others.

The Abstract Task Graph (ATaG) framework is a data-driven macroprogramming paradigm in which the application developer specifies the system behavior in a mixed declarative-imperative manner. The interactions between various components of a system are specified as a *task graph*, while the internal working of the tasks themselves are specified in an imperative language like Java. The ATaG tasks are annotated with *instantiation* and *firing* rules, which specify the location (e.g., *one per floor*) and

*This work is partially supported by the European Union under the IST-004536 RUNES project and by the National Science Foundation, USA, under grant number CCF-0430061.

triggering constraint (e.g., *when the data is available*) of each task. The channels connecting the tasks to data items are also annotated to specify the physical range of data they are interested in (e.g. *gather data from a 30 m radius*). More details of ATaG can be found in [2]. The compiled ATaG programs run atop a runtime system (DART), which abstracts the underlying sensor network as a distributed data store, with which the tasks interact only with `getData()`and `putData()` primitives. DART also takes care of activities such as task firing, routing and data item management. More details of DART can be found in [1].

In this work, we present our *end-to-end* solution for macroprogramming WSNs consisting of Sun SPOTs using ATaG's data-driven programming paradigm. As described in detail in Section 4, the application developer needs to provide **a)** the description of the WSN application in ATaG, and **b)** details about the target system deployment. The toolkit assists the user in specifying the above, and then our compiler generates the code to be deployed on each node in the system. Out toolkit also deploys the resulting customized node-level Java code to each Sun SPOT.

2 Sun SPOTs

A Sun SPOT node [8] has a 180 MHz 32 bit ARM920T processor with 512K RAM and 4M Flash memory. Its extensible sensor board also contains a 2G/6G 3-axis accelerometer, a temperature sensor, and a light sensor. The SPOTs run the Squawk Java virtual machine directly out of flash memory, and can run programs written using Java2ME libraries.

Although the SPOTs are slightly bulkier than the commonly used much smaller Tmote sky nodes, their higher processing speed and larger memory make them more capable, at roughly the same cost. Since users can write programs for these nodes in Java, they do not have to learn programming in nesC/TinyOS. This is greatly beneficial for designers of system level software for SPOT based WSNs, since they do not need to learn a new language. The Sun SPOTs are especially of interest to us since the architecture of our DART [1] runtime system assumes the existence of a pre-emptive scheduler, which is provided by the use of threads in Java. Finally, although other WSN platforms exist on which Java programs can be run, the fact that the JVM runs directly on the metal in the SPOTs reduces the overheads involved.

3 Class of Application

Macroprogramming using ATaG is not intended for lower level tasks such as routing. ATaG's data driven paradigm can be used to develop a wide class of applications using complex interactions patterns such as *hierarchical data gathering*, *localized interactions* and *actuation driven by sensing*. This fact was illustrated in [7], by giving examples of how ATaG can be used to design application such as Landslide Detection, Target Tracking, and HVAC (heating, ventilation and air-conditioning) Management. In the next section, we discuss how our toolkit can be used to design and deploy WSN application, by taking a sense-and-respond application as an example.

4 Programming SunSPOTs using ATaG: Case Study

4.1 Toolkit and Design Flow

Figure 1 depicts the workflow an application developer will go through when developing WSN applications using our toolkit. The left side of the figure depicts the framework, whereas the right side depicts the implementation details of our current system. Note that the framework itself is generic, and can be ported to work with other types of WSN nodes as well.

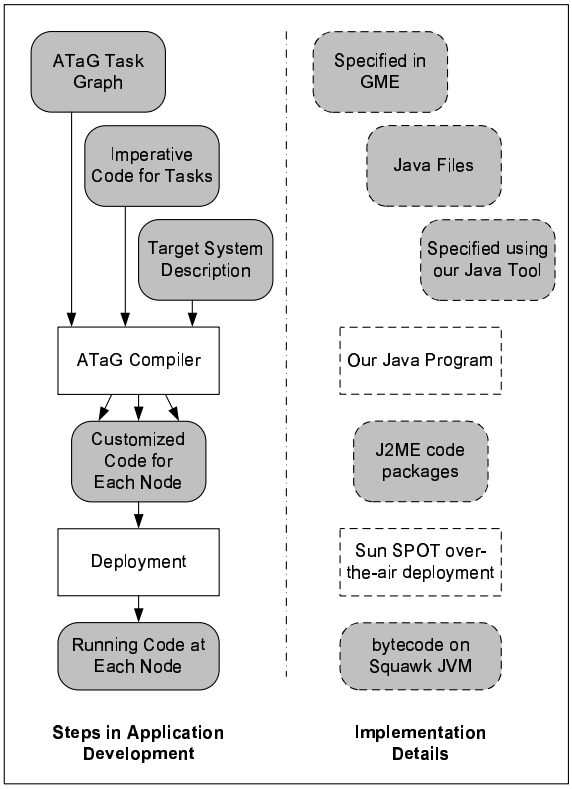


Figure 1. Workflow using Toolkit

To specify a WSN application to be deployed on a network of SunSPOTs, the application developer first specifies the **task graph** using the Generic Modeling Environment (GME) [3], as shown in Figure 2(b). We have developed a meta-model of ATaG in GME, and the tasks and data items are both treated as models in GME. As shown in the figure, the user can draw the entire task graph, as well as specify attributes of tasks and channels using our toolkit. The **imperative part** of each abstract task is specified as a Java program. We have provided standard interfaces for the programmer to **a)** interface with the sensors and the LEDs on the SPOTs, as well as for **b)** reading and writing data items to the runtime system. Since ATaG programs are platform-independent, the developer needs to provide the **target system description** (*viz.* node locations, radio range, region labels, and sensing/actuation capabilities) separately. As part of our toolkit, we have also developed a Java-based application into which developers can upload the system description and visualize/edit it.

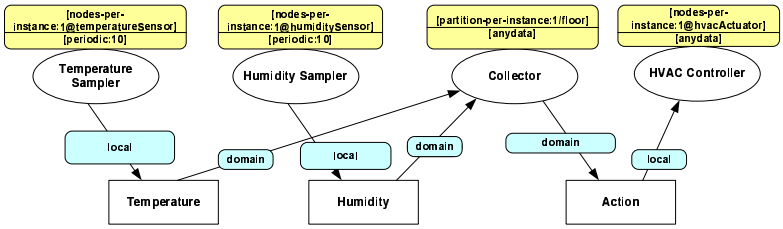
Once the user specifies the ATaG program using our GME metamodel, our program written using the pattern-based generator-interpretor of GME converts it into an equivalent XML representation. The ATaG compiler (written in Java) takes this XML and the system description, and computes the task assignment for each node in the system. Additionally, it also computes estimates of the expected cost of running the system in terms of the number of messages passed etc. More details can be found in [6]. The compiler outputs a set of Java2ME files for each node, including the customized runtime system. The runtime system includes the protocols for routing data items across nodes. Note that in this case, the abstraction of the target architecture used by the compiler is the Squawk JVM running on the SPOTs. Our toolkit also generates the appropriate ant scripts, which when executed, deploys the corresponding code to each node using standard over-the-air (OTA) mechanisms provided by Sun SPOTs. Note that only single-hop OTA deployment is possible in the current version of Sun SPOTs.

Example Application. Figure 2(a) shows the task graph for the HVAC application described in [7]. Figure 2(b) depicts the same task graph, as expressed in our toolkit, and highlights its various panels.

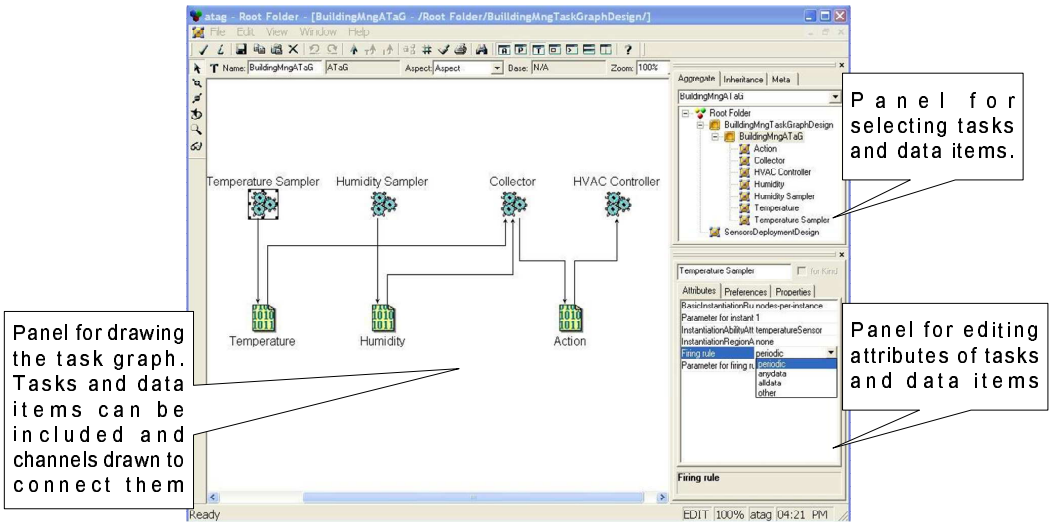
5 Conclusion

Our end-to-end toolkit can be used by developers to create a wide range of WSN applications, starting from a clear idea of the tasks constituting the system, and ending with an actual working system comprised of state-of-the art networked sensing nodes. Although we discuss only one application in this work, we intend to demonstrate the full power and ease of use of our tool at the conference.

We believe that our toolkit, by providing an easy to use end-to-end platform to develop WSN applications on the Sun SPOT nodes, will enable the development of complex WSN applications. We intend to make this tool available for download and use the feedback provided by the community to further our research in the area of macroprogramming WSNs.



(a) As ATaG Task Graph



(b) In our Environment

Figure 2. WSN Application for Building HVAC Management

References

[1] A. Bakshi, A. Pathak, and V. K. Prasanna. System-level support for macroprogramming of networked sensing applications. In *Int. Conf. on Pervasive Systems and Computing (PSC)*, 2005.

[2] 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)*, June 2005.

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

[4] R. Gummadi, O. Gnawali, and R. Govindan. Macro-programming wireless sensor networks using Kairos. In *Proc. of the* 1^{st} *Int. Conf. on Distributed Computing in Sensor Systems (DCOSS)*, June 2005.

[5] R. Newton and M. Welsh. Region streams: Functional macroprogramming for sensor networks. In *Proc of the* 1^{st} *Int. Workshop on Data Management for Sensor Networks (DMSN)*, 2004.

[6] A. Pathak, L. Mottola, A. Bakshi, G. P. Picco, and V. K. Prasanna. A compilation framework for macroprogramming networked sensors. In *Proc. of the the* 3^{rd} *Int. Conf. on Distributed Computing on Sensor Systems (DCOSS)*, 2007.

[7] A. Pathak, L. Mottola, A. Bakshi, V. K. Prasanna, and G. P. Picco. Expressing sensor network interaction patterns using data-driven macroprogramming. In *Proc. of the* 3^{rd} *Int. Wkshp. on Sensor Networks and Systems for Pervasive Computing (PerSens - colocated with IEEE PERCOM)*, 2007.

[8] Sun^{TM} Small Programmable Object Technology (Sun SPOT), `www.sunspotworld.com`.

[9] K. Whitehouse, J. Liu, and F. Zhao. Semantic streams: a framework for composable inference over sensor data. In *Third European Workshop on Wireless Sensor Networks (EWSN)*, February 2006.