This Page Intentionally Left Blank

ARCHITECTURE-INDEPENDENT PROGRAMMING FOR WIRELESS SENSOR NETWORKS
This Page Intentionally Left Blank
ARCHITECTURE-INDEPENDENT PROGRAMMING FOR WIRELESS SENSOR NETWORKS

Amol B. Bakshi
University of Southern California

Viktor K. Prasanna
University of Southern California
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>xi</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>xv</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td></td>
</tr>
<tr>
<td>1.1 Sensor networks and traditional distributed systems</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Programming of distributed sensor networks</td>
<td>7</td>
</tr>
<tr>
<td>1.2.1 Layers of programming abstraction</td>
<td>7</td>
</tr>
<tr>
<td>1.2.1.1 Service-oriented specification</td>
<td>7</td>
</tr>
<tr>
<td>1.2.1.2 Macroprogramming</td>
<td>8</td>
</tr>
<tr>
<td>1.2.1.3 Node-centric programming</td>
<td>10</td>
</tr>
<tr>
<td>1.2.2 Lessons from parallel and distributed computing</td>
<td>12</td>
</tr>
<tr>
<td>1.3 Macroprogramming: What and why?</td>
<td>14</td>
</tr>
<tr>
<td>1.4 Contributions and outline</td>
<td>16</td>
</tr>
</tbody>
</table>
2 The Abstract Task Graph

2.1 Target applications and architectures 21
2.2 Key concepts 23
  2.2.1 Data-driven computing 23
    2.2.1.1 Program flow mechanisms 23
    2.2.1.2 Why data-driven? 26
  2.2.2 Mixed imperative–declarative specification 28
2.3 Syntax 29
  2.3.1 The Structure of an ATaG program 29
  2.3.2 More on task annotations 34
  2.3.3 Illustrative examples 39
2.4 Semantics 45
  2.4.1 Terminology 45
  2.4.2 Firing rules 46
  2.4.3 Task graph execution 48
  2.4.4 get(·) and put(·) 48
2.5 Programming idioms 49
  2.5.1 Object tracking 51
  2.5.2 Interaction within local neighborhoods 52
  2.5.3 In-network aggregation 52
  2.5.4 Hierarchical data fusion 55
  2.5.5 Event-triggered behavior instantiation 56
2.6 Future work 59
  2.6.1 State-based dynamic behaviors 59
  2.6.2 Resource management in the runtime system 61
  2.6.3 Utility-based negotiation for task scheduling and resource allocation 63

3 DART: The Data-Driven ATaG Runtime 65

3.1 Design objectives 65
  3.1.1 Support for ATaG semantics 65
  3.1.2 Platform independence 66
  3.1.3 Component-based design 67
  3.1.4 Ease of software synthesis 68
3.2 Overview 69
3.3 Components and functionalities 72
3.3.1 Task, data, and channel declarations 72
3.3.2 UserTask 75
  3.3.2.1 Service 75
  3.3.2.2 Interactions 75
  3.3.2.3 Implementation 77
3.3.3 DataPool 79
  3.3.3.1 Service 79
  3.3.3.2 Interactions 79
  3.3.3.3 Implementation 79
3.3.4 AtagManager 82
  3.3.4.1 Service 82
  3.3.4.2 Interactions 82
  3.3.4.3 Implementation 83
3.3.5 NetworkStack 87
  3.3.5.1 Service 87
  3.3.5.2 Interactions 87
  3.3.5.3 Implementation 87
3.3.6 NetworkArchitecture 88
  3.3.6.1 Service 88
  3.3.6.2 Interactions 88
  3.3.6.3 Implementation 89
3.3.7 Dispatcher 90
  3.3.7.1 Service 90
  3.3.7.2 Interactions 91
  3.3.7.3 Implementation 91
3.4 Control flow 93
  3.4.1 Startup 94
  3.4.2 get() and put() 97
  3.4.3 Illustrative example 100
3.5 Future work 101
  3.5.1 Lazy compilation of channel annotations 101
  3.5.2 Automatic priority assignment for task scheduling 102
4 Programming and Software Synthesis 105

4.1 Terminology 106

4.2 Meta-modeling for the ATaG domain 106
   4.2.1 Objectives 106
   4.2.2 Application model 108
   4.2.3 Network model 110

4.3 The programming interface 112

4.4 Compilation and software synthesis 115
   4.4.1 Translating task annotations 117
   4.4.2 Automatic software synthesis 117
   4.4.3 The ATaG simulator 121
   4.4.4 Initialization 122
      4.4.4.1 Situatedness 123
      4.4.4.2 Network interface 124
      4.4.4.3 Network architecture 124
      4.4.4.4 Sensor interface 124
   4.4.5 Visualizing synthesized application behavior 128

5 Case Study: Application Development with ATaG 135

5.1 Overview of the use case 136

5.2 Designing the macroprograms 136
   5.2.1 Temperature gradient monitoring 136
   5.2.2 Object detection and tracking 139

5.3 Specifying the declarative portion 142

5.4 Imperative portion: Temperature gradient monitoring 143
   5.4.1 Abstract data items: Temperature and fire 143
   5.4.2 Abstract task: Monitor 144
   5.4.3 Abstract task: Temperature sampler 151
   5.4.4 Abstract task: Alarm actuator 153

5.5 Imperative portion: Object detection and tracking 155
   5.5.1 Abstract data items: TargetAlert and TargetInfo 155
   5.5.2 Abstract Task: SampleAndThreshold 157
   5.5.3 Abstract Task: Leader 157
   5.5.4 Abstract Task: Supervisor 165

5.6 Application Composition 165

5.7 Software Synthesis 171
PREFACE

Networked sensing is an area of enormous research interest, as evidenced by the explosive growth of technical workshops, conferences, and journals related to topics in sensor networks as well as by the increasing number of related book publications. Research in sensor networks is influenced to varying degrees by ideas from traditional parallel and distributed computing, wireless ad hoc networking, signal processing, information theory, and so on. The semantics of spatial computing applications in sensor networks necessitate enhancements and extensions to traditional ideas in some cases and require the development of entirely new paradigms in others. The next generation of context-aware applications for these systems will require novel phenomenon-centric programming models, methodologies, and design tools to translate high-level intentions of the programmer into executable specifications for the underlying deployment. Indeed, such tools are critical for further development of the field; and once they become available, dramatic growth in this field can be expected.
This book deals with macroprogramming of networked sensor systems. A "macro"-programming language allows the application developer to express program behaviors at a high level of abstraction. The job of translating this high-level specification into node-level behaviors is delegated to a compilation and software synthesis system. Macroprogramming is interesting because it promises to facilitate rapid application development for large-scale, possibly heterogeneous sensor networks and also provides a framework for optimizing task placement and communication in such networks, without user involvement.

Objectives

We present a methodology and a programming language—called the Abstract Task Graph (ATaG)—for architecture-independent macroprogramming of networked sensor systems. Architecture-independence allows applications to be developed prior to decisions being made about the network deployment and also allows the same application to be compiled onto different target deployments.

ATaG is built upon two fundamental concepts: (1) the use of data-driven computing as the underlying control flow mechanism and (2) the adoption of mixed imperative-declarative notation for program specification. We argue that the former enables modular, composable programs for sensor networks and also provides an intuitive paradigm for specifying reactive behaviors in networked sensing. The latter separates concerns of task placement, firing, and in-network communication from the actual application functionality and is the key to architecture independence.

The objective of this book is to illustrate the feasibility and usefulness of architecture-independent programming for networked sensor systems. The discussion is centered around the ATaG model, which is discussed in detail. Ultimately, we want the reader to gain exposure to the high-level concepts that guided the design and implementation of the ATaG programming language and environment. We also discuss the implementation of the DART runtime system in great detail. This is because we want the reader to be familiar not just with the broad outline of DART but with its intimate details that will enable him/her to modify and/or extend the DART functionality as desired. Eventually, it is our hope that researchers can build upon ATaG and DART and design full-fledged compilation and code synthesis environments for a variety of networked sensor systems.
**Book Organization**

Chapter 1 provides a brief overview of sensor networks and the differences between sensor networks and traditional distributed systems. Various layers of programming abstraction for networked sensor systems are also reviewed, and the motivation for macroprogramming is discussed.

Chapter 2 presents the Abstract Task Graph (ATaG) model. A discussion of the ATaG syntax and semantics is followed by a section on programming idioms in ATaG. ATaG programs for oft-cited behaviors in networked sensing (hierarchical tree structures, object tracking, etc.) are presented.

Chapter 3 discusses the design of DART the Data-driven ATaG RunTime. An overview of the DART components is followed by an in-depth discussion of each component. Relevant code listings from the current implementation of DART accompany the discussion.

Chapter 4 outlines the overall process of application development with ATaG. This includes the graphical programming interface for ATaG, the automatic software synthesis mechanism, and the rudimentary compiler that translates ATaG programs into node-level behaviors. The simulation and visualization interface for ATaG is also discussed.

Chapter 5 presents an ATaG case study. In this chapter, we illustrate programming and synthesis of a composite application consisting of a gradient monitoring component and an object tracking component. We walk the reader through the steps involved in developing the declarative and imperative parts of the ATaG program and the software synthesis and rudimentary compilation support offered by the programming environment.

Chapter 6 concludes this book by discussing the broader context of the ATaG research. We argue that ATaG is not just a specific language for a class of sensor network applications but also a general framework that can be extended to a variety of behaviors in current and future sensor network applications. ATaG is also a framework for compilation in the sense that the syntax and semantics of ATaG and the design of the DART runtime system provide a well-defined framework for "intelligent compilation" of sensor network applications for a variety of target architectures.

**Target Audience**

This book is written for (i) researchers in networked embedded sensing and pervasive computing, (ii) researchers in parallel and distributed computing with applications to context-aware spatial computing, (iii) practitioners
involved in implementing and deploying networked sensor systems, and (iv) application developers and software engineers for networked embedded systems for pervasive computing.

We particularly hope that the in-depth discussion of the design of the runtime system and of the simulation and visualization environment will enable interested researchers to download the software and use it to demonstrate extensions of the programming model or of the runtime system itself. To this end, we discuss specific extensions to ATaG and DART as future work in various clearly marked sections of this book.

AMOL B. BAKSHI
VIKTOR K. PRASANNA

Los Angeles, California
January, 2008
ACKNOWLEDGMENTS

The ATaG programming model and the associated programming environment was born in the summer of 2004 during the author Amol Bakshi’s internship at the Palo Alto Research Center. Special thanks are due to Jim Reich and Dan Larner for co-inventing the programming model, as well as to Maurice Chu, Qingfeng Huang, Patrick Cheung, and Julia Liu for patient hearings and constructive feedback during the course of the summer work. We thank Prof. Ramesh Govindan and Prof. Bhaskar Krishnamachari (USC) for contributing broad perspectives on wireless networked sensing, specific inputs on the strengths and weaknesses of the ATaG model, and suggestions for future work.

We are grateful to Animesh Pathak and Qunzhi Zhou at the University of Southern California for their wholehearted adoption and ongoing furtherance of the ATaG research on macroprogramming for sensor networks. Finally, an enormous amount of gratitude is due to the wonderful—and wonderfully patient—people who encouraged this book project and drove it to completion: Prof. Albert Zomaya (Founding Editor-in-Chief of the Wiley Book Series on Parallel and Distributed Computing), Whitney Lesch, Val Moliere, Paul Petralia, Emily Simmons, Lisa Morano Van Horn, and Anastasia Wasko (Wiley); and Amy Hendrickson (TlExnology, Inc.).

A.B.B.
V.K.P.

xv