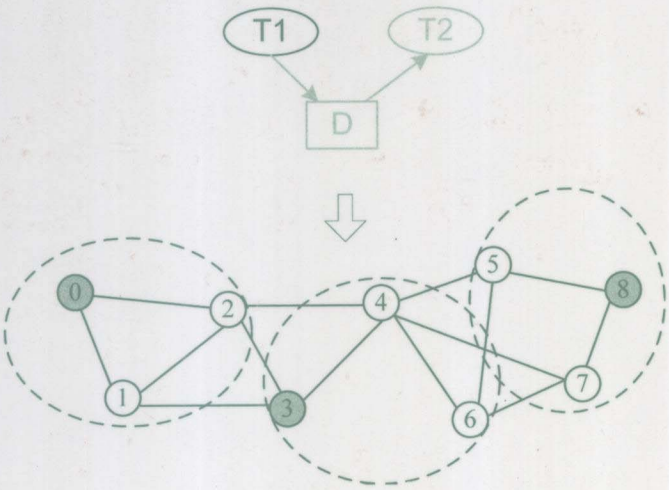

WILEY

ARCHITECTURE-INDEPENDENT PROGRAMMING FOR WIRELESS SENSOR NETWORKS

AMOL B. BAKSHI
VIKTOR K. PRASANNA

Series on Parallel and Distributed Computing • Series Editor, Albert Y. Zomaya

ARCHITECTURE-INDEPENDENT PROGRAMMING FOR WIRELESS SENSOR NETWORKS

Amol B. Bakshi
University of Southern California

Viktor K. Prasanna
University of Southern California

WILEY-INTERSCIENCE

A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey
Published simultaneously in Canada

Library of Congress Cataloging-in-Publication Data:

Bakshi, Amol B., 1975-
Architecture-independent programming for wireless sensor networks / Amol B. Bakshi, Viktor K. Prasanna.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-471-77889-9 (cloth)
1. Sensor networks--Programming. 2. Wireless LANs--Programming. I. Prasanna Kumar, V. K. II. Title.
TK7872.D48 B
681'.2--dc22

2007046862

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

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 (TEXnology, Inc.).

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

CONTENTS

CHAPTER 1

INTRODUCTION

Networked sensor systems

A networked sensor system (a "sensor network") is a distributed computing system where some or all nodes are capable of interacting with the physical environment. These nodes are termed as sensor nodes and the interaction with the environment is through sensing interfaces. Sensors typically measure properties such as temperature, pressure, humidity, flow, etc., when sampled. The sensed value can be one-dimensional or multi-dimensional. Sensor networks have a wide range of applications. Acoustic sensing can be used to detect and track targets in the area of deployment. Temperature, light, humidity, and motion sensors can be used for effective energy management through climate moderation in homes and commercial buildings.

Wireless sensor networks (WSNs) [44, 3, 17] are a new class of sensor networks, enabled by advances in VLSI technology and comprised of sensor nodes with small form factors, a portable and limited energy supply, on-board sensing, computing, and storage capability, and wireless connectivity through

By Amol B. Bakshi, Viktor K. Prasanna

a bidirectional transceiver. WSNs promise to enable dense, long-lived embedded sensing of the environment. The unprecedented degree of information about the physical world provided by WSNs can be used for in situ sensing and actuation. WSNs can also provide a new level of context awareness to other back-end applications, making sensor networks an integral part of the vision of pervasive, ubiquitous computing—with the long-term objective of seamlessly integrating fine grained sensing infrastructure into larger, multi-tier systems.

There has been significant research activity over the last few years in the system-level aspects of wireless sensing. System level refers to the problems such as: (a) localization [41] and time synchronization [15, 16] to provide the basic "situatedness" for a sensor node node; (b) energy-efficient medium access protocols that aim to increase the system lifetime through means such as coordinated sleep–wake scheduling [60]; (c) novel routing paradigms such as geographic [33, 47], data-centric [22], and trajectory-based [40] that provide the basic communication infrastructure in a network where the assignment and use of globally unique identifiers (such as the IP addresses of the Internet) is infeasible or undesirable; (d) modular, component-based operating systems for extremely resource constrained nodes [27], etc. A variety of routing and data fusion protocols for generic patterns such as multiple-source single-sink data gathering trees are also being developed to optimize for a range of goodness metrics [30, 29, 61]. A comprehensive overview of state of the art in system level aspects of wireless embedded sensing can be found in [31, 18].

1.1 SENSOR NETWORKS AND TRADITIONAL DISTRIBUTED SYSTEMS

It is instructive to compare and contrast the fundamental nature of networked sensing with traditional parallel and distributed computing, with a view to identifying the degree to which the research in the latter field over the past few decades can be leveraged (with our without modification) to propose solutions for analogous problems in the former. Since the primary focus of this work is on models and methodologies for programming of large-scale networked sensor systems, the comparison will be biased towards aspects which influence application development and not so much on system level issues.

Sensor networks are essentially collections of autonomous computing elements (sensor nodes) that pass messages through a communication network and hence fit the definition of a distributed computing system proposed in [8]. However, some of the fundamental differences between networked sensor systems and traditional distributed computing systems are as follows:

Transformational versus reactive processing

The primary reasons for programming applications for a majority of traditional distributed computing systems were "high speed through parallelism, high reliability through replication of process and data, and functional specialization" [8]. Accordingly, the objective of most programming models and languages was to (i) allow the programmer to expose parallelism for the compiler and runtime system to exploit and (ii) provide support for abstractions such as shared memory that hide the distributed and concurrent nature of the underlying system from the application developer. In other words, the purpose of most abstractions was to allow the programmer to still visualize the target architecture as a von Neumann machine, which provided an intuitive and straightforward mental model of reasoning about sequential problem solving. Alternate approaches such as dataflow and functional programming were also proposed, motivated by a belief in the fundamental unsuitability of the von Neumann approach for parallel and distributed computing [5]. Regardless of the approach, most parallel and distributed applications were ultimately transformational systems that are characterized by a function that maps input data to output data. This function can be specified as a sequential, imperative program for a von Neumann architecture, and the purpose of parallelizing and distributing the execution over multiple nodes is mainly to reduce the total latency.

A networked sensor system is not a transformational system that maps a well-defined set of input data to an equally well-defined set of output data. Instead, like a majority of embedded software, it is a continuously executing and primarily reactive system that has to respond to external and internal stimuli [24]. An event of interest in the environment triggers computation and communication in the network. A quiescent environment ideally implies a quiescent network as far as application level processing is concerned.

Space awareness

An embedded sensor network can be considered to represent a discrete sampling of a continuous physical space. In fact, an abstract model of a distributed sensor network can be defined and analyzed purely in terms of measurements of the space being monitored [32], without any reference to the network architecture. In contrast to traditional distributed computing where all compute nodes were basically interchangeable and the physical location of a particular computing element is not directly relevant from a programming or optimization perspective, space awareness [63] is an integral part of embedded net-

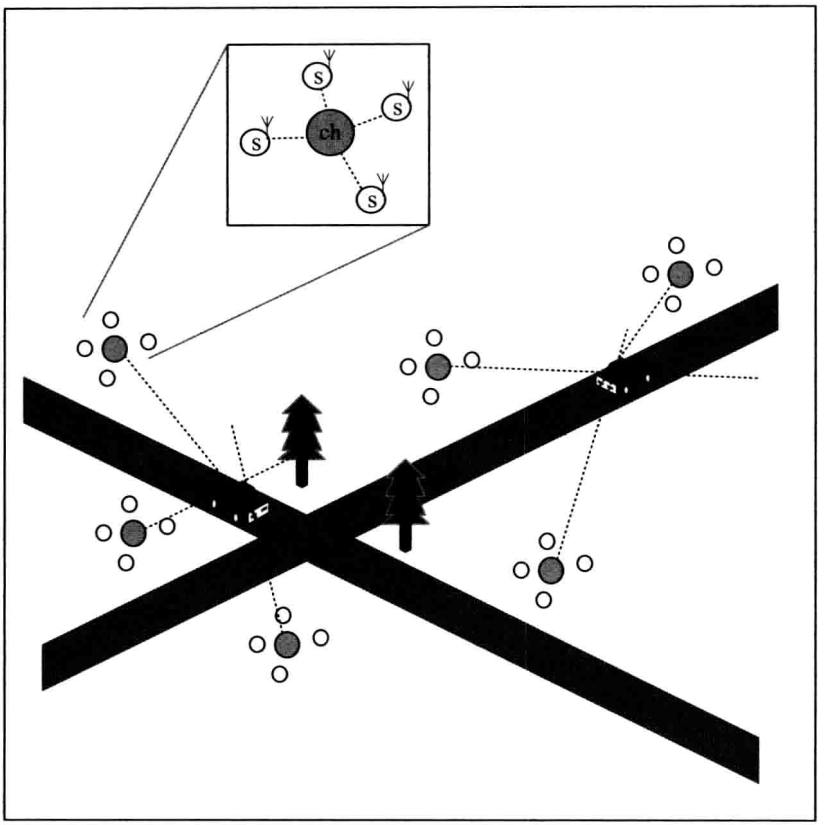

Figure 1.1 An example sensor network deployment for vehicle detection and tracking. Sensor nodes are deployed in clusters, with each cluster consisting of a relatively powerful clusterhead node and four resource-constrained sensor nodes. Each sensor could be equipped with acoustic and/or magnetic sensors. The individual sensor nodes in each cluster communicate their readings to the clusterhead which computes the line of bearing and possibly the type of vehicls. This information will be relayed to a supervisor station that can triangulate the object position by ending line of bearing estimates from multiple clusters. This particular scenario was one of the early use cases for wirelessly networked sensor systems.