Chapter 14
Event-Driven Mobile Computing with Objects

Tom Van Cutsem
Vrije Universiteit Brussel, Belgium

Wolfgang De Meuter
Vrije Universiteit Brussel, Belgium

ABSTRACT

We motivate why event-driven approaches are suitable to address the challenges of mobile and ubiquitous computing. In particular, we describe the beneficial properties of event-based communication in so-called mobile ad hoc networks. However, because contemporary programming languages feature no built-in support for event-driven programming, programmers are often forced to integrate event-driven concepts with a different programming paradigm. In particular, we study the difficulties in combining events with the object-oriented paradigm. We argue that these difficulties form the basis of what we call the object-event impedance mismatch. We highlight the various issues at the software engineering level and propose to resolve this mismatch by introducing a novel object-oriented programming language that supports event-driven abstractions from the ground up.

INTRODUCTION

This chapter focuses on programming abstractions for mobile computing (Mascolo, Capra, & Emmerich, 2002), a research domain that studies Weiser’s vision of ubiquitous computing (Weiser, 1991) from a distributed systems’ perspective. Mobile computing applications are deployed on mobile devices (e.g. cellular phones, PDAs, …) equipped with wireless communication technology (e.g. WiFi, Bluetooth,…). Such devices form so-called mobile ad hoc networks, which are characterized by the fact that connectivity between devices is often intermittent (connections drop and are restored as devices physically move about) and the fact that there is little or no fixed support infrastructure, such that devices can often communicate only with physically proximate devices.

Event-based coordination is a natural fit for such networks. Events can be disseminated to multiple nearby interested parties (subscribers) without necessarily knowing the exact identity...
of these subscribers. This key property of event-based systems is crucial in a ubiquitous computing context, where the identity and number of nearby devices is not known at development time.

Contemporary software is not built using event-driven abstractions from the ground up. Rather, software development is predominantly object-oriented. In this paradigm, distributed applications are expressed in terms of distributed objects sending messages to one another. Thus, an application programmer using a mainstream object-oriented language will be forced to implement his or her own event infrastructure on top of the object-oriented infrastructure.

We will describe that combining object technology with event-based technology is not without problems. In a nutshell, the most important differences are the following. Objects communicate by means of messages, not by means of events. They do so via remote object references, which can only refer to a single object throughout their lifetime. Using an event-based system, however, a publisher can send messages to an arbitrary number of subscribers. On the other hand, objects introduce useful abstractions such as request/response interactions that cannot be directly expressed using pure event-based publish/subscribe communication. We have named the combination of these and a number of other issues the object-event impedance mismatch, by analogy with the object-relational impedance mismatch which describes the difficulties in combining objects with relational databases for the purpose of persistence (Carey & DeWitt, 1996).

We resolve the object-event impedance mismatch by means of a novel object-oriented programming language named AmbientTalk (Van Cutsem, Mostinckx, Gonzalez Boix, Dedecker, & De Meuter, 2007) that supports event-based programming from the ground up. For example, AmbientTalk provides asynchronous event notification between objects as a primitive operation. Rather than using a traditional multithreaded concurrency model, the language features a reactive event loop concurrency model (Miller, Tribble, & Shapiro, 2005). Also, the language supports referencing abstractions that allow a single object to directly refer to an entire group of distributed, proximate objects. Messages sent via such references are automatically treated as events that are broadcast to all objects subscribed to the group.

After having briefly introduced AmbientTalk’s main concepts, we show how the language enables one to program in an object-oriented yet event-driven way thus overcoming the object-event impedance mismatch.

The objectives of this chapter are threefold. Our goal is to:

1. give a detailed understanding of why event-driven communication is highly suitable in mobile ad hoc networks.
2. discuss the key differences between event-driven and object-oriented programming, and the consequences thereof for the developer.
3. propose a novel programming model that is both object-oriented and event-driven, allowing developers to use the strengths of both models without unnecessary complications raised by the object-event impedance mismatch.

BACKGROUND

Many systems designed specifically for mobile ad hoc networks (MANETs) adopt an event-driven communication paradigm. We will support this claim by discussing a number of concrete event-driven middleware systems designed for MANETs. Before doing so, we first describe why event-driven communication is so useful in MANETs by showing how it promotes loose coupling between communicating parties.