Alan Dearle
Susan Eisenbach (Eds.)

LNCS 3798

Component Deployment

Third International Working Conference, CD 2005
Grenoble, France, November 2005
Proceedings

Springer

TP311.5-53
C737
2005

Alan Dearle Susan Eisenbach (Eds.)

Component Deployment

Third International Working Conference, CD 2005
Grenoble, France, November 28-29, 2005
Proceedings

Springer

Volume Editors

Alan Dearle
University of St Andrews, School of Computer Science
North Haugh, St Andrews, Fife KY16 9SX, UK
E-mail: al@dcs.st-andrews.ac.uk

Susan Eisenbach
Imperial College London, Department of Computing
180 Queens Gate, London, SW7 2BZ, UK
E-mail: s.eisenbach@imperial.ac.uk

Library of Congress Control Number: 2005936342

CR Subject Classification (1998): D.2, F.3, D.1, D.3, D.4

ISSN	0302-9743
ISBN-10	3-540-30517-3 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-30517-0 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11590712 06/3142 5 4 3 2 1 0

Preface

This volume of *Lecture Notes in Computer Science* contains the proceedings of the 3rd Working Conference on Component Deployment (CD 2005), which took place from 28 to 29, November 2005 in Grenoble, France, and co-located with Middleware 2005. CD 2005 is the third international conference in the series, the first two being held in Berlin and Edinburgh in 2002 and 2004, respectively. The proceedings of both these conferences were also published by Springer in the *Lecture Notes in Computer Science* series and may be found in volumes 2370 and 3083.

Component deployment addresses the tasks that need to be performed after components have been developed and addresses questions such as:

- What do we do with components after they have been built?
- How do we deploy them into their execution environment?
- How can we evolve them once they have been deployed?

CD 2005 brought together researchers and practitioners with the goal of developing a better understanding of how deployment takes place in the wider context. The Program Committee selected 15 papers (12 long papers, three short papers) out of 29 submissions. All submissions were reviewed by at least three members of the Program Committee. Papers were selected based on originality, quality, soundness and relevance to the workshop.

We would like to thank the members of the Program Committee (Mikio Aoyama, Noureddine Belkhatir, Judy Bishop, Paul Brebner, Wolfgang Emmerich, Thomas Gschwind, Richard Hall, Andre van der Hoek, Nenad Medvidovic, Andrea Polini and Peter Sewell) for providing timely and significant reviews, and for their substantial effort in making CD 2005 a successful workshop.

We would also like to thank the following additional reviewers: Doug Palmer, Sam Malek, Chris Mattmann, Andrew J. McCarthy, Marija Mikic-Rakic, Chiyoung Seo and Rob Chatley for their assistance in reviewing papers.

The CD 2005 submission and review process was supported by the Cyber Chair Conference Management System. We are indebted to the services of Borbola Online Conference Services and in particular Richard van de Stadt for their excellent support in managing this system. Andrew J. McCarthy must also be thanked for his diligent efforts in collating the papers in these proceedings.

The workshop was held in conjunction with Middleware 2005. We would like to acknowledge the help from the Middleware 2005 Organizing Committee for their assistance, during the organization of CD 2005, in creating this co-located event.

We would also like to acknowledge the prompt and professional support from Springer, who published these proceedings in printed and electronic volumes as part of the *Lecture Notes in Computer Science series*.

September 2005

Alan Dearle
Susan Eisenbach

Organization

Program Committee

Program Chairs

- Alan Dearle
  University of St Andrews, UK
  al@dcs.st-and.ac.uk
- Susan Eisenbach
  Imperial College, London, UK
  sue@doc.ic.ac.uk

Program Committee Members

- Mikio Aoyama
  Network Information and Software Engineering Laboratory, Japan
  mikio.aoyama@nifty. com
- Noureddine Belkhatir
  IMAG LSR, Grenoble, France
  Noureddine.Belkhatir@imag. fr
- Judy Bishop
  University of Pretoria, South Africa
  jbishop@cs.up.ac.za
- Paul Brebner
  CSIRO ICT Centre, Canberra, Australia
  Paul.Brebner@csiro.au
- Wolfgang Emmerich
  University College London, UK
  w.emmerich@cs.ucl.ac.uk
- Thomas Gschwind
  Technische Universität Wien, Austria
  thomasg@ieee.org
- Richard Hall
  IMAG LSR, Grenoble, France
  heavy@ungoverned.org
- Andre van der Hoek
  University of California, Irvine, USA
  andre@ics.uci.edu
- Nenad Medvidovic
  University of Southern California, Los Angeles, USA
  neno@usc.edu

- Andrea Polini
  CNR, Pisa, Italy
  andrea.polini@isti.cnr.it
- Peter Sewell
  University of Cambridge, UK
  Peter.Sewell@cl.cam.ac.uk
- Kurt Wallnau
  Carnegie Mellon University, Pittsburgh, USA
  kcw@sei.cmu.edu
- Alexander Wolf
  University of Lugano, Switzerland
  alexander.wolf@unisi.ch

¥396.48元

Table of Contents

Cooperative Component-Based Software Deployment in Wireless Ad Hoc Networks

Hervé Roussain and Frédéric Guidec

University of South Brittany, France
{Herve.Roussain, Frederic.Guidec}@univ-ubs.fr

Abstract. This paper presents a middleware platform we designed in order to allow the deployment of component-based software applications on mobile devices (such as laptops or personal digital assistants) capable of ad hoc communication. This platform makes it possible to disseminate components based on peer-to-peer interactions between neighboring devices, without relying on any kind of infrastructure network. It implements a cooperative deployment scheme. Each device runs a deployment manager, which maintains a local component repository, and which strives to fill this repository with software components it is missing in order to satisfy the deployment requests expressed by the user. To achieve this goal the deployment manager continuously interacts in the background with peer managers located on neighboring devices, providing its neighbors with copies of software components it owns locally, while obtaining itself from these neighbors copies of the components it is looking for.

1 Introduction

The number and variety of lightweight mobile devices capable of wireless communication is growing significantly. Such devices include laptops, tablet PCs, personal digital assistants (PDAs), many of which are now shipped with built-in IEEE 802.11 (a.k.a. Wi-Fi [1]) network interfaces. With such interfaces, the devices can occasionally be connected to an infrastructure network, using so-called access points that play the role of gateways. But the 802.11 standard also makes it possible for mobile devices to communicate directly in ad hoc mode, that is, without relying on any kind of infrastructure network. An ad hoc network is thus a network that can appear and evolve spontaneously as mobile devices themselves appear, move and disappear dynamically in and from the network [9].

For the users of laptops or PDAs, the prospect of deploying software applications on these devices as and when needed obviously appears as an attractive one, no matter if these devices communicate in infrastructure or in ad hoc mode. Yet, solutions for component-based software deployment have been proposed mostly for infrastructure-based environments so far, while very little effort has been devoted to software deployment in purely ad hoc networks.

In this paper we describe a model we devised in order to allow the deployment of component-based software applications on mobile devices participating in an ad hoc network. In Section 2 we motivate our approach by showing how infrastructure-based networks and ad hoc networks constitute radically different environments as far as software deployment is concerned, and we show that solutions that prove efficient in infrastructure environments are hardly applicable in ad hoc environments. In Section 3 we present CODEWAN (COmponent DEployment in Wireless Ad hoc Networks), a middleware platform that implements our model. The main characteristics of this platform are discussed in Section 4, which also lists some directions we plan to work along in the future. In Section 5 we compare CODEWAN with other works that also address the problem of software deployment, either in infrastructure environments, or in ad hoc environments. Section 6 concludes the paper.

2 Motivations

In this section we show that deploying software components in an ad hoc network raises issues that usually do not appear in infrastructure networks. As a reminder, we first describe how software component provision and delivery are commonly performed in an infrastructure-based environment. We then show that an ad hoc network presents additional constraints that need to be addressed specifically.

2.1 Software Deployment in an Infrastructure Network

Figure 1 illustrates a typical infrastructure network, including stable and mobile hosts—typically, workstations and laptops—interconnected through gateways (such as routers and switches). In such an environment some of the stable hosts can be in charge of storing components in so-called *component repositories*, and of implementing server programs capable of delivering these components on demand. Other hosts in the network can then behave as simple clients with respect to these servers. Whenever the owner—or the administrator—of one of the client hosts initiates the deployment of a new component-based software application on this device, the problem mostly comes down to locating the server—or servers—capable of providing the components required by this application, and downloading these components so they can be installed locally.

Consider the example shown in Figure 1, and assume that the owner of device *A* decides to initiate the installation on this device of an application that requires components *c1*, *c2* and *c3*. The deployment middleware running on device *A* must first identify

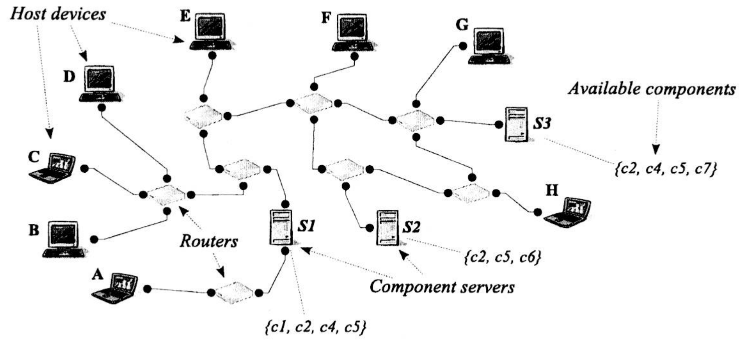

Fig. 1. Illustration of software component deployment in an infrastructure network

one or several servers capable of delivering these components. A component may actually be provided by several servers, for example in order to balance the workload in the network, or to allow fault tolerance. In any case, once a client has identified a server that can provide a component, obtaining this component simply requires its download between the server and the client. Note that in such a context the deployment of a component on a given host can usually be considered as a "real time" operation: once a user has ordered the deployment middleware to locate and download a component, this operation can usually be performed immediately. In the remaining of this section, we show that deploying components in an ad hoc environment can in contrast require a more lengthy process, which requires some middleware capable of enforcing a deployment strategy in the background on behalf of the user.

2.2 Software Deployment in a Dynamic Ad Hoc Network

Figure 2 shows a typical dynamic ad hoc network, which consists of a collection of portable communicating devices. The devices in such a network are usually highly mobile and volatile. Device mobility results from the fact that each device is carried by a user, and users themselves move quite a lot. Device volatility is the consequence of the fact that, since the devices usually have a limited power-budget, they are frequently switched on and off by their owners.

A major characteristic of wireless ad hoc networks is that communication interfaces have a limited transmission range. Consequently any device can only communicate directly with neighboring devices. Multi-hop transmissions can sometimes be obtained by implementing a dynamic routing algorithm on each device [10,13], but it is worth observing that even with dynamic routing, a realistic ad hoc network often presents itself as a fragmented network. Such a network appears as a—possibly changing—collection of so-called "islands" (also referred to as "clouds" or "cells" in the literature). Mobile devices that belong to the same island can communicate together, using either multi-hop or single-hop transmissions depending on whether dynamic routing is used or not in the network. However, devices that belong to distinct islands cannot communicate together, because no transmission is possible between islands.

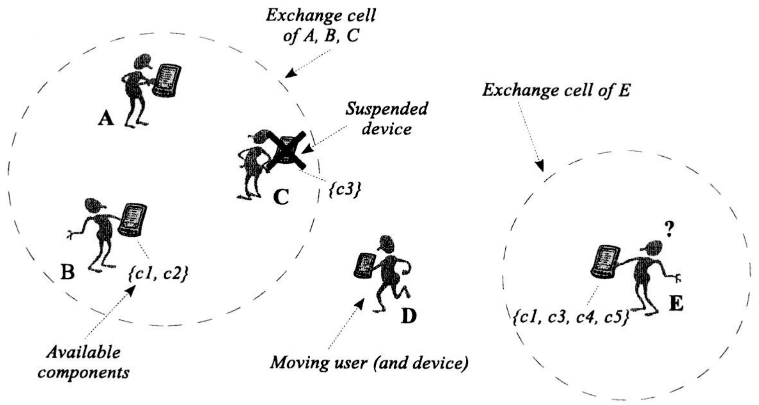

Fig. 2. Illustration of software component deployment in a dynamic ad hoc network

In such a context, a traditional client-server deployment scheme such as that illustrated in Section 2.1 is hardly applicable, as no device is stable and accessible enough to play the role of a server of components, maintaining a component repository and allowing client devices to access this repository whenever needed.

In the remainder of this paper, we present a model we propose in order to allow for these constraints. Basically, instead of being able to access a server whenever needed, each device must maintain a local component repository. A peer-to-peer interaction model then makes it possible for a device to cooperate with its neighborhood, by allowing its neighbors to obtain copies of the software components available on its local repository, while itself benefiting from a similar service offered by its neighbors.

Consider the example shown in Figure 2, and assume again that the owner of device *A* wishes to install on this device an application that requires components *c1*, *c2* and *c3*. In our example, *A* can obtain components *c1* and *c2* from device *B*. But as devices *C* and *E*—that both own a copy of component *c3*—are (possibly temporarily) unreachable, *A* cannot obtain a copy of component *c3* from any of these devices. Yet *A* could obtain component *c3* from device *C* if this device was switched on by its user. It could also obtain this component from device *E* if *A*'s user happened to walk towards *E*, or if *E*'s user happened to walk toward *A*. A roaming device such as *D* may even serve as a benevolent carrier between *E* and *A*, transporting component c3—and possibly other components as well—between separate islands, and thus contributing to the dissemination of software components and applications all over the network.

This example shows that when the owner of a mobile device participating in an ad hoc network requests the deployment of a component-based application on this device, there is no guarantee that this request can be satisfied immediately, as there is no guarantee that the components required for this deployment are readily accessible in the neighborhood. Yet, since the topology of an ad hoc network can change continuously and unpredictably as devices move and are switched on or off, the fact that a given component cannot be obtained at a given time does not involve that this component will remain inaccessible in the future. There is thus a need for some deployment middleware capable of ensuring the collection of missing components in the background in order to satisfy the user's needs.

3 Towards Software Component Deployment on Mobile Devices

In this section, we present an overview of CODEWAN (*COmponent DEployment in Wireless Ad hoc Networks*), a platform we designed in order to support the deployment of component-based software applications on mobile devices communicating in ad hoc mode. CODEWAN implements a cooperative model, where neighboring devices interact in order to discover and exchange software components. Each device implements a local component repository, and a deployment manager is responsible for maintaining this repository on behalf of the user. Any component stored in the repository can be used to assemble and start an application locally. Copies of this component can also be sent on demand to neighboring devices.

3.1 Overview of the CODEWAN Platform

The platform is built as a three-layer model, as shown in Figure 3. The upper and lower layers in this model have been described in details in [7] and [3] respectively. They are thus only described briefly below, and the paper then continues with a detailed description of the model's central layer, which implements the component repository and the deployment manager that maintains this repository.

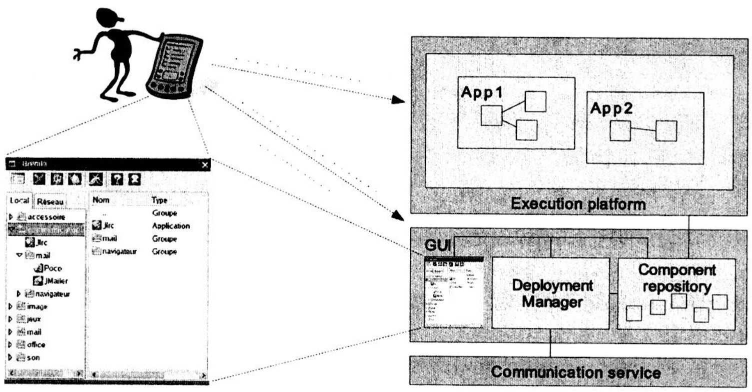

Fig. 3. Overview of the CODEWAN platform and screenshot of its GUI on a PDA

The upper layer in the platform is meant to provide a framework for assembling and running applications. Instead of defining its own component-model, CODEWAN was designed so as to rely on existing execution frameworks for component-oriented or service-oriented applications. In its current implementation it interfaces with JAMUS, a runtime framework that is primarily dedicated to hosting potentially malicious mobile applications [7], as well as with JULIA, an execution framework for applications designed using the Fractal component model [12].

The lower layer in our model was designed in order to support the asynchronous dissemination of so-called *transfer documents* in an ad hoc network. A transfer document is an XML document whose external element's attributes specify the conditions required for disseminating the document in the network. These attributes thus play approximately the same role as header fields in IP packets or in UDP datagrams. They indicate typically the document's source and destination, the expected propagation scope for this document, etc.

The "payload" of a transfer document consists of the internal XML elements that are embedded in the document. Any kind of structured information can be transported in a transfer document. In CODEWAN, though, transfer documents are used to transport software package descriptors in the network.

Figure 4 shows a typical transfer document. Attributes in this document indicate that it was sent by device *shiva*, and that it was addressed to any device in the neighborhood (notice that the communication layer CODEWAN relies on supports the use of wildcard addresses). The payload in this transfer document consists of a package descriptor, whose role and structure are detailed in Section 3.3.

```
<transfer-document
  document-id="fb54356fe468d9"
  source="device:shiva"  destination="device:*"
  hops-to-live="3"  lifetime="01:00:00"
  service-type="package-advertisement">
  <package-descriptor>
     <general-information
       type="application/java"  category="communication/messaging"
       name="JMessager"  version="1.3"
       provider="Laboratoire  Valoria"
       summary="JMessager is a P2P messager"/>
     <java-application name="masc.jmessager.JMessagerImpl" />
     <dependencies>
       <required-package  name="JMessengerUI" version="1.2"/>
       <required-package  name="P2PAsyncDissemination"/>
       <optional-package  name="AddressBook" version="2.0"/>
     </dependencies>
   </package-descriptor>
</transfer-document>
```

Fig. 4. Example of an XML transfer document carrying a software package descriptor

The communication layer in CODEWAN provides services for encapsulating transfer documents in UDP datagrams. Large XML documents can be fragmented and then transported in distinct, smaller transfer documents that each can fit in a single UDP datagram. The communication layer of course supports the re-assembly of such fragments after they have been received from the network. Documents can be transferred either in unicast, broadcast, or multicast mode, and using either single-hop or multi-hop transmissions. In the latter case, all mobile devices in the network are expected to behave as routers, using algorithms for dynamic routing and flooding such as those described in [13,11,14].

Further details about CODEWAN's communication layer can be found in [3]. In the remainder of this paper we focus on the description of the central layer of the platform. The deployment manager is implemented in this layer, together with the component repository this manager is in charge of maintaining. The repository is a place where software components can be stored locally on a mobile device. Components stored in this repository are thus readily available for the execution framework that constitutes the upper layer of the platform. The deployment manager takes orders from the user, and interacts with peer managers that reside on neighboring devices in order to fill the local repository with components required by the user, while providing its peers with components they need in order to satisfy their own users.

3.2 Installation Steps in CODEWAN

The deployment manager can provide the user with information about all the applications it knows about. At any time a given application is either:

- *installed locally* (meaning that this application is either already running in the local execution framework, or ready to be loaded and started in this framework);
- *installable* (meaning that all the components required for running this application are available in the local repository, so the application could be installed immediately if the user requested it);