Electronic Communications of the EASST
Volume 11 (2008)

Proceedings of the
First International DisCoTec Workshop on
Context-aware Adaptation Mechanisms for
Pervasive and Ubiquitous Services
(CAMPUS 2008)

Learning-based Coordination of Distributed Component Deployment

Yves Vanrompay Peter Rigole Yolande Berbers

6 pages

Guest Editors: Romain Rouvoy, Mauro Caporuscio, Michael Wagner
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Learning-based Coordination of Distributed Component Deployment

Yves Vanrompay **Peter Rigole** **Yolande Berbers**

Department of Computer Science, K.U.Leuven
Celestijnenlaan 200A,
3001 Leuven, Belgium
{Yves.Vanrompay, Peter.Rigole, Yolande.Berbers}@cs.kuleuven.be

Abstract: Self-organizing and resource-aware component deployment is an important feature of mobile pervasive systems. Distributed resources must be dynamically allocated to software components to ensure QoS demands and not distracting the user. In this paper, we propose a Reinforcement Learning technique to optimize distributed component deployment and migration. We argue that the approach meets some main requirements demanded by applications running on mobile systems. A motivating scenario is presented in which a distributed application server allows users to share content and run applications in mobile ad-hoc networks.

Keywords: Distributed component deployment, Reinforcement Learning

1 Introduction

The vision of autonomic computing [2] leads to self-organizing systems that deploy their components taking into account current resource availability. A pervasive computing environment is mainly composed of resource-constrained mobile devices. Mobility also means that user needs on the one hand and availability of memory and bandwidth on the other hand are varying. Therefore it is a challenge to adapt and deploy applications to optimize offered QoS while minimizing user distraction. We propose a distributed adaptation reasoning and decision making approach to deploy a set of application components in a distributed, optimal and resource-aware manner on a set of nodes (Figure 1). Components (*C1* to *C5*) forming applications (*S1* to *S7*) are distributed on a number of nodes *n1* to *n4* in the environment to make a balanced use of available resources. At runtime, components can be redeployed or migrated in case of a change in resources. The planning algorithm for distributed component deployment (DiComPloy) is based on Collaborative Reinforcement Learning (CRL) [4] and takes into account the cost of instantiating and executing components. We focus primarily on communication and computation cost in terms of bandwidth and memory since these are critical resources in a mobile environment.
This paper is organized as follows. The next section gives an overview of the requirements we derived as important for adaptive mobile systems. Also, a real-life scenario motivating the need for resource-aware distributed component deployment is presented. Section 3 explains how we applied the learning algorithm as an optimization strategy for component deployment. The next section evaluates the proposed mechanism against the requirements and scenario. Section 5 compares our proposal with related work. Finally, we draw some conclusions and elaborate on ongoing and future work.

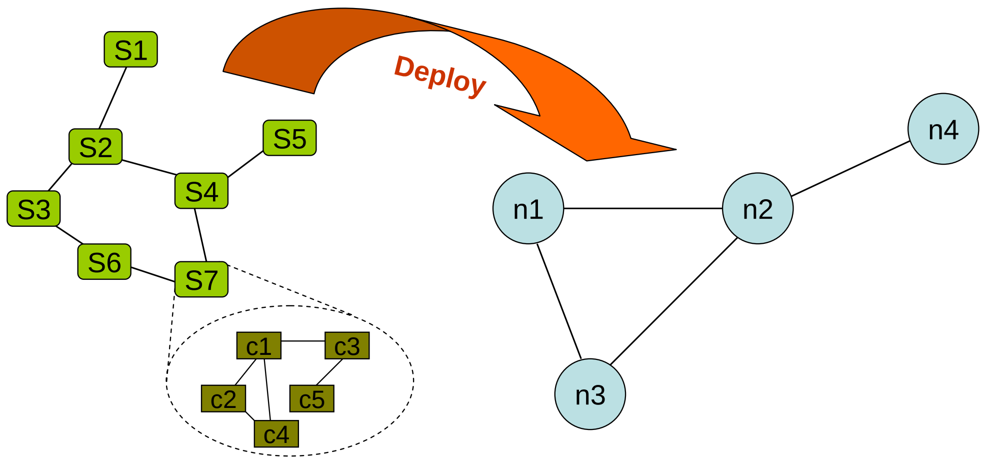

Figure 1: Distributed component deployment

2 Requirements and motivating scenario

Our work focuses on a middleware system [7] which supports the development and deployment of context-aware, adaptive applications. With the emphasis on distributed adaptation and deployment mechanisms, we identified the following requirements:

- Adaptation mechanisms and the associated strategies should be reusable and evolvable in large set of applications, contexts, and users.[R01]
- A ubiquitous planning process should take into account the dynamic dimension of ubiquitous environments.[R02]
- Component deployment and migration should take into account the limited resources on targeted mobile devices.[R03]
- Component deployment and migration should consider limited or costly bandwidth situations and should support mobile agent mechanisms to offload costly tasks.[R04]
- AI planning can be applied during the reconfiguration of an application to divide the reconfiguration process and delegate reconfiguration tasks.[R05]
- The modelling approach should provide a support for resource-based quality contracts.[R06]
- Models used at run-time should include QoS properties of both software and hardware components.[R07]

In the following, we describe a scenario showing how mobile, context-aware adaptive applications may need distributed component deployment and migration. The scenario also motivates the requirements we discussed above.

We envision a distributed platform for sharing content, gaming, chat, etc. formed by devices joining a federation in an ad-hoc manner. Users perceive an *InstantSocial* cloud as an ordinary site offering services. However, these services are scattered across nearby devices and integrated by a distributed application server running on top of our middleware. The server consists of a dynamic composition of components like an ontology server, a video server, a presentation layer and so on. This composition is distributed and replicated which means the distributed server is a combination of different implementations on nearby devices forming an ad-hoc network. Users get notified of topics they are possible interested in or multiplayer games that can be joined. DiComPloy is used to adapt and configure the ad-hoc server infrastructure needed for *InstantSocial*. Server components can be deployed, redeployed or migrated across the different available devices taking into account the resource availability of each individual device. Users can also specify a maximum amount of resources they want to reserve for the distributed server components. While devices come and leave and the ad-hoc network evolves, components can be dynamically redeployed to ensure the QoS for the users, balance the use of resources and minimize user distraction. Whenever fixed infrastructure is available, this can be used as a backup and for deployment components that use a lot of resources. A way of using *InstantSocial* is on metro trips where instant sites could appear out of nothing just by travelers having their applications turned on. Other usages include conferences, concerts, festivals and sports events.

3 Collaborative Reinforcement Learning

CRL is an unsupervised learning method that derives a way of behaving from interaction with an uncertain environment. It discovers which actions produce the best results by trying them. The algorithm learns to choose the actions that optimize the rewards in the longer term. Actions that give a poor immediate pay-off may be taken in the anticipation of a higher return in the longer term. Action selection is probabilistic and depends on a trade off between exploration and exploitation. Exploitation means that actions are preferred that are deemed to result in an optimal behaviour, while exploration ensures that the learned experience covers all actions. This combination is a powerful way to deal with dynamic environments that are subject to frequent changes in their behaviour. CRL enables collaborating hosts to solve optimization problems in dynamic decentralized networks, in this case optimal component deployment. The deployment mechanism is controlled by the middleware and transparent for the application. Components can be deployed remotely and relocated at runtime in order to respect the resource quality requirements. Thus, the deployment is resource-driven, concentrating on memory and bandwidth. DiComPloy is a decentralized component choreography approach. Choreography indicates the shared coordination technique across multiple instances of the middleware to develop a solution for a resource-aware deployment problem. DiComPloy is able to:

- Find an optimal deployment: Deployment of the optimal amount of functionality whenever possible on the host that matches the deployment's resources needs best.
- Perform redeployment: The ability to change the existing deployment when the configuration of the computing environment changes with respect to resource availability.
- Exhibit self-organizing deployment behaviour: User distraction should be avoided in the

deployment process. Cooperating middleware instances need to be able to agree upon distributed application deployment.

A cost function is introduced that takes into account memory and bandwidth, and can be extended with other resources. The cost for a successful deployment of a component on a node is the cost of the memory needed by the component on that host (depending on the current available amount of memory on that host) and the bandwidth cost in case the component has to be migrated to that node. The advantage of specifying a cost function for each host is that the weights for each resource dimension in the cost formula can be host-dependent. The bandwidth cost on a host operating on battery power, for example, will be much higher than the bandwidth cost on a host that is powered by the power net.

4 Evaluation

As this paper discusses some early results of our research, this section investigates whether and how the proposed approach satisfies the requirements and scenario needs that were identified in Section 2. The first requirement refers to scalability and evolvability in large sets of applications, contexts and users. DiComPloy is a scalable and self-organizing way to deal with component deployment and migration. Applications can be distributed over a potentially large set of nodes belonging to different users. DiComPloy is also generic in the sense that it can take into account different types of context information. Not only memory and bandwidth can be part of the cost function, but the function can be extended with battery power, type of network connectivity, user preferences, etc. The cost function can be part of a more general utility function. The current version of DiComPloy optimizes deployment with respect to memory and bandwidth, which are especially limited resources on mobile devices [R03]. Deploying a large component on a device with little memory will be more costly than deploying it on a device with large memory [R04]. Models used at run-time include QoS properties of both software and hardware components [R07]. Both the QoS that can be provided by a node as the resources needed by a component are used in the decision process due to the cost function. Resource-based quality contracts optionally can be incorporated in the coordination process since memory and bandwidth cost can be expressed as a contract a host has to adhere to for the component deployment to be optimal[R06]. The balance of exploration and exploitation of CRL takes into account the dynamics of a ubiquitous environment [R02]. Changes in the hardware environment are detected by exploration after which a redeployment or migration of components can take place. DiComPloy enables collaborating hosts to reconfigure the deployment setting and delegate reconfiguration tasks amongst them since it is by nature a decentralized coordination technique [R05].
The *InstantSocial* scenario clearly shows the need for distributed component deployment in a real-life application. An application server that is distributed in an evolving mobile ad-hoc network consisting of resource-constrained hosts must be (re)deployed in a self-organizing way. DiComPloy provides a mechanism to orchestrate efficient deployment (re)configuration.

5 Related work

Classic techniques for distributed adaptation and deployment typically use a centralized approach. Brute force algorithms explore the whole search space and evaluate all alternative configurations [8]. While they guarantee to find an optimal deployment, they do not support scalability to a large number of variants and nodes.
Mikic-Rakic et al. [6] have developed a centralized approach to deploy components. While the approach is shown to work well for small deployment problems, it does not scale well to large problems. Moreover, only memory and network connection reliability are taken as criteria for deployment cost. Scholz et al. [1] propose a Divide and Conquer (DnC) approach to achieve a balanced and resource-aware deployment of multiple applications that is flexible in case of changes in the device topology. Large applications are divided into smaller units (packs) that can be adapted and deployed separately on available resources. DiComPloy is complementary to DnC. By first applying DnC applications can be divided into packs. After this optional clustering step, DiComPloy can be used to deploy the packs in an optimal way. Rigole [5] presents a decentralized component choreography approach based on CRL that is extended to solve multidimensional optimization problems in a distributed way. It is shown that this is a fully scalable technique to deploy components on a set of nodes. Dowling et al. [3] evaluate CRL as a technique that enables groups of agents to solve optimization problems online in dynamic decentralized networks. They have implemented CRL in a routing protocol for mobile ad-hoc networks and argue it is a suitable approach for environments where topology, resources and node availability are subject to frequent and unpredictable changes.

6 Conclusions and future work

In this paper, we presented a self-organizing learning technique to optimize distributed component (re)deployment in a dynamic environment. This optimization problem is solved in a distributed way by coordination amongst the several nodes. We argued how DiComPloy meets requirements that are important for mobile adaptive systems and motivated this with an example application. Decentralization prevents the adaptation mechanism from having a single point of failure and represents a scalable approach to distributed deployment of applications on nodes.
A code base that works in a simulation environment currently exists, but has yet to be deployed on and experimented with in a real-life distributed environment. Future work includes the adaptation and integration of the existing code into the middleware platform, after which a thorough evaluation can be carried out. While for the moment we only take into account memory and bandwidth to decide on optimal deployment, we plan to extend our approach to other resources like battery power and user preferences.

Acknowledgements: The authors of this paper would like to thank their partners in the MUSIC-IST project and acknowledge the financial support given to this research by the European Union (6th Framework Programme, contract number 35166).

References

[1] Scholz U., Rouvoy R., Divide and Conquer - Scalability and Variability for Adaptive Middleware In: Proceedings of the International Workshop on the Engineering of Software Services for Pervasive Environments (ESSPE'07), pp. 35-39, Dubrovnik, Croatia, ACM. AICPS.

[2] Kephart J., Chess D., The Vision of Autonomic Computing, Computer Magazine, Jan. 2003

[3] Dowling J., Curran E., Cunningham R., Cahill V., Using feedback in collaborative reinforcement learning to adaptively optimise manet routing. IEEE Transactions on Systems, Man and Cybernetics (Part A), Special issue on Engineering Self-Organized Distributed Systems, 35(3): 360-372, May 2005.

[4] J. Dowling. The Decentralised Coordination of Self-Adaptive Components for Autonomic Distributed Systems. PhD thesis, Trinity College, Dublin, Ireland, October 2004.

[5] P. Rigole. Task- and Resource-Aware Component Deployment in Ambient Intelligence Environments. PhD thesis, K.U.Leuven, Leuven, Belgium, November 2006.

[6] M. Mikic-Rakic and N. Medvidovic. Architecture-level support for software component deployment in resource constrained environments. In Proceedings of the First International IFIP/ACM Working Conference on Component Deployment (CD02), Berlin, Germany, June 2002.

[7] http://www.ist-music.eu

[8] IST MADAM project. Theory of Adaptation. Deliverable 2.2. December 2006. p. 4449. www.ist-madam.org