Chapter 10

Niche:
A Platform for Self-Managing Distributed Applications

Vladimir Vlassov  
*KTH Royal Institute of Technology, Sweden*

Ahmad Al-Shishtawy  
*KTH Royal Institute of Technology, Sweden*

Per Brand  
*Swedish Institute of Computer Science, Sweden*

Nikos Parlavantzas  
*Université Européenne de Bretagne, France*

**ABSTRACT**

We present Niche, a general-purpose, distributed component management system used to develop, deploy, and execute self-managing distributed applications. Niche consists of both a component-based programming model as well as a distributed runtime environment. It is especially designed for complex distributed applications that run and manage themselves in dynamic and volatile environments. Self-management in dynamic environments is challenging due to the high rate of system or environmental changes and the corresponding need to frequently reconfigure, heal, and tune the application. The challenges are met partly by making use of an underlying overlay in the platform to provide an efficient, location-independent, and robust sensing and actuation infrastructure, and partly by allowing for maximum decentralization of management. We describe the overlay services, the execution environment, showing how the challenges in dynamic environments are met. We also describe the programming model and a high-level design methodology for developing decentralized management, illustrated by two application case studies.

DOI: 10.4018/978-1-60960-845-3.ch010
INTRODUCTION

Autonomic computing (Horn, 2001) is an attractive paradigm to tackle the problem of growing software complexity by making software systems and applications self-managing. Self-management, namely self-configuration, self-optimization, self-healing, and self-protection, can be achieved by using autonomic managers (IBM, 2006). An autonomic manager continuously monitors software and its execution environment and acts to meet its management objectives. Managing applications in dynamic environments with dynamic resources and/or load (like community Grids, peer-to-peer systems, and Clouds) is especially challenging due to large scale, complexity, high resource churn (e.g., in P2P systems) and lack of clear management responsibility.

This chapter presents the Niche platform (Niche, 2010) for self-managing distributed applications; we share our practical experience, challenges and issues, and lessons learned when building the Niche platform and developing self-managing demonstrator applications using Niche. We also present a high-level design methodology (including design space and steps) for developing self-managing applications.

Niche is a general-purpose, distributed component management system used to develop, deploy, and execute self-managing distributed applications or services in different kinds of environments, including very dynamic ones with volatile resources. Niche is both a component-based programming model that includes management aspects as well as a distributed runtime environment.

Niche provides a programming environment that is especially designed to enable application developers to design and develop complex distributed applications that will run and manage themselves in dynamic and volatile environments. The volatility may be due to the resources (e.g., low-end edge resources), the varying load, or the action of other applications running on the same infrastructure. The vision is that once the infrastructure-wide Niche runtime environment has been installed, applications that have been developed using Niche, can be installed, and run with virtually no effort. Policies cover such issues as which applications to scale down or stop upon resource contention. After deployment the application manages itself, completely without human intervention, excepting, of course, policy changes. During the application lifetime the application is transparently recovering from failure, and tuning and reconfiguring itself on environmental changes such as resource availability or load. This cannot be done today in volatile environments, i.e., it is beyond the state-of-the-art, except for single machine applications and the most trivial of distributed applications, e.g., client/server.

The rest of this chapter is organized as follows. The next section lays out the necessary background for this work. Then, we discuss challenges for enabling and achieving self-management in a dynamic environment characterized by volatile resources and high resource churn (leaves, failures and joins of computers). Next, we present Niche. We provide some insight into the Niche design ideas and its architecture, programming model and execution environment, followed by a presentation of programming concepts and some insight into the programming of self-managing distributed applications using Niche illustrated with a simple example of a self-healing distributed group service. Next, we present our design methodology (including design space and design steps) for developing a management part of a self-managing distributed application in a decentralized manner, i.e., with multiple interactive autonomic managers. We illustrate our methodology with two demonstrator applications, which are self-managing distributed services developed using Niche. Next, we discuss combining a policy-based management (using a policy language and a policy engine) with hard-coded management logic. Finally, we present some conclusions and our future work.