Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Original Software Publication

The Siebog multiagent middleware

Dejan Mitrović[a], Mirjana Ivanović[a], Milan Vidaković[b,*], Zoran Budimac[a]

[a] Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Serbia
[b] Faculty of Technical Sciences, University of Novi Sad, Serbia

A R T I C L E   I N F O

Article history:
Received 21 July 2015
Revised 23 February 2016
Accepted 22 March 2016
Available online 14 April 2016

Keywords:
Multiagent middleware
Software agents
Java EE
Html5

A B S T R A C T

This paper presents Siebog, a software framework and an execution environment for the development of software agents. Built using the standard *Java EE* and *HTML5* technologies, it provides all the benefits of clustered computing on the server, as well as platform-independence on the client. The tight integration of the two development paradigms has resulted in a system that also provides heterogeneous agent mobility, cross-platform messaging, and code sharing.

Table 1
Software metadata.

Nr.	(executable) Software metadata description	Please fill in this column
S1	Current software version	1.3.1
S2	Permanent link to executables of this version	https://github.com/ElsevierKnowledgeBasedSystems/KNOSYS-D-15-01003
S3	Legal software license	Apache license 2.0
S4	Computing platform/Operating system	Linux, Windows, OS X
S5	Installation requirements & dependencies	Java version 8.x or greater
S6	If available, link to user manual - if formally published include a reference to the publication in the reference list	[6]. See [1,8,9] for descriptions of XJAF and Radigost.
S7	Support email for questions	mitrovic.dejan@gmail.com

Table 2
Code metadata.

Nr.	Code metadata description	Please fill in this column
C1	Current code version	1.3.1
C2	Permanent link to code/repository used of this code version	https://github.com/ElsevierKnowledgeBasedSystems/KNOSYS-D-15-01003
C3	Legal code license	Apache license 2.0
C4	Code versioning system used	git
C5	Software code languages, tools, and services used	Java EE, JavaScript, Eclipse IDE for Java EE developers, Ant
C6	Compilation requirements, operating environments & dependencies	Java. All dependencies (JAR files) are included in the project.
C7	If available Link to developer documentation/manual	https://github.com/gcvt/siebog/wiki
C8	Support email for questions	mitrovic.dejan@gmail.com

* Corresponding author. Tel.: +381214852422.
E-mail addresses: mitrovic.dejan@gmail.com (D. Mitrović), mira@dmi.uns.ac.rs (M. Ivanović), minja@uns.ac.rs (M. Vidaković), zjb@dmi.uns.ac.rs (Z. Budimac).

http://dx.doi.org/10.1016/j.knosys.2016.03.017
0950-7051/© 2016 Elsevier B.V. All rights reserved.

1. Introduction

Software agents (or simply, *agents*) can rarely operate on their own; often, they need a runtime environment (also called a *multiagent middleware*), to support their execution. In this paper we present Siebog, our multiagent middleware designed for the modern web. Its main purpose is to provide infrastructural support for multiagent systems with practical applications in many artificial intelligence domains, including swarm intelligence, artificial (social) life, distributed machine learning and decision making, etc.

The main advantage of Siebog is that it achieves better performances and uses more advanced technologies than other existing agent middlewares. For example, to achieve agent load-balancing and fault-tolerance, we use ready-made industrial solutions, instead of devising proprietary solutions. Therefore, Siebog can run twice as many agents as other frameworks [1] on the same hardware. We have also introduced JavaScript-based agents that work in web browsers, that can communicate with server-based agents and can migrate to the server, if needed.

Over the years, a lot of research effort has been dedicated to the development of multiagent middlewares [2]. Agent researchers have recognized this ongoing trend of migrating software to the web and have provided web access to their respective solutions. But, this is usually done in an inefficient manner. For example, in many Java-based solutions (e.g., JADE [3] and JaCaWeb [4]) the web component is provided in form of a Java applet. But Java applets require a browser plug-in, which is unavailable on many platforms or requires manual installation. With one of its parts implemented in pure JavaScript, Siebog supports a wider range of hardware and software configurations.

The support for distributed execution is present in almost all existing agent middlewares. However, most of them use plain computer networks and/or implement their own approaches for agent load-balancing and fault-tolerance (e.g., [3,5]). One disadvantage of these approaches is lower flexibility. For example, in JADE the agent developer needs to manually specify which agent is hosted on which computer, while in Siebog this process is performed automatically [6].

The goal of Siebog is to provide an infrastructure for executing agents in web environments, but in accordance to the modern standards. We demonstrate that it is not necessary to “reinvent the wheel.” Instead, it is more beneficial to use existing, standards-compliant, and well-tested solutions offered by Java EE and HTML5.

The only other purely HTML5-based agent platform that we are aware of is described in [7]. However, Siebog uses more advanced client-side JavaScript technologies to achieve true multi-threading and full-duplex communication. Similarly, on the server our system provides more advanced clustering features described earlier.

Section 2 of this paper discusses the Siebog middleware in details, including its functionalities and implementation. An illustrative example is presented in Section 3 while the overall conclusions are given in Section 4.

2. Siebog multiagent middleware

Siebog is based on our previous two systems, *Extensible Java EE-based Agent Framework* (XJAF) [1,8] and *Radigost* [9], (Table 1) in a way that not only combines their individual functionalities but also results in new features.

The overall architecture is shown in Fig. 1. Client-side agents can be executed on a range of devices, such as smartphones, tablets, desktop computers, and Smart TVs. They use a JavaScript library to access the framework’s functionalities, and can communicate with the server over AJAX or the WebSocket protocol.

The server side of Siebog is executed on top of computer clusters. It consists of several modules called *managers* [1]. The *Agent Manager* is in charge of controlling the agents’ life-cycles. The *Message Manager* is in charge of inter-agent communication, while the *WebClient* acts as a directory of remote client-side Siebog instances.

If needed, each client-side agent can have its own *stub* representation on the server. To other entities, the stub appears as a regular server-side agent. However, it simply forwards all incoming messages to the actual client-side instance. Therefore, Siebog can achieve completely transparent communication between agents located in distributed heterogeneous client-side devices (demonstrated in Section 3).

2.1. Software functionalities

On the server side, the two most important features of Siebog are *scalability* and *fault-tolerance*. Scalability allows our system to automatically distribute agents across the cluster [1]. This feature enables Siebog to support large agent societies. Even with the same number of machines in cluster, our middleware can operate twice the number of agents compared to the JADE [1]. Fault-tolerance is concerned with replicating the state of each server-side component and restoring it in case of software or hardware failures.

The main advantage of the Siebog’s client-side is platform independence [9], a greater one than provided by Java. Being a web application, Siebog also requires no installation or configuration steps. And finally, as demonstrated in [9], its runtime performance is comparable to that of a desktop multiagent platform: our JavaScript-based agents executed complex tasks in web browsers in approximately 25 ms, while JADE-based Java agents executed the same tasks in approximately 20 ms.

The tight integration of client and server sides of Siebog has resulted in several new important functionalities. First of all, agents are able to communicate with each other regardless of their “physical” locations. A lot of effort has also been invested in providing agent code sharing, which means that an agent can be written only once and executed both on the server and on the client. Finally, heterogeneous agent mobility allows agents to freely move between the client and the server.

2.2. Implementation details

The server side of Siebog is implemented in Java. Most of its functionalities stem from Java EE 7 components and the WildFly application server.

Agents are realized as *Enterprise JavaBean* (EJB) components. In the vast majority of cases, stateful session EJBs are used [1]. Agent communication is achieved using the *Java Message Service* (JMS), which takes care of important issues such as message ordering, delivery in case of failures, etc.

Server-side managers are realized as stateless session EJBs. This offers the greatest flexibility and runtime performance, since a stateless EJB can freely be created on any cluster node. To share information across the cluster (e.g., the details about running agents), managers rely on the distributed *Infinispan* cache included in the WildFly server.

The client-side of Siebog is implemented in pure JavaScript. Agents are represented as Web workers, so each one has its own (native) thread of execution. Web workers also provide a communication infrastructure, allowing agents in a single web page to directly exchange messages.

3. Illustrative examples

The Siebog’s ability to run large agent societies was demonstrated in [1]. The experiment included pairs of agents. In each pair, the first agent would issue a request to the second, which