Eve: A Multi-Agent Approach to an Open-Source and Web-Based Platform

Desa Avxhi
Informatics Department
Faculty of Natural Sciences
University of Tirana
desa.avxhi@fshnstudent.info

Areti Bojaxhiu
Informatics Department
Faculty of Natural Science
University of Tirana
areti.bojaxhiu@fshn.edu.al

Abstract

In recent years, systems based on multi-agents are developing faster. Yet, the difficulty of achieving the communication between agents in heterogeneous environments is still a problem. In this paper, we will give a closer look at Eve, a new platform that has recently been developed. It aims to be open and dynamic and makes it very easy to use the concept of software agents. Two most important characteristics of this platform are its accessibility and its ease of implementation for any developer in any development environment.

1. Introduction

In recent years, there has been a rapidly growing interest in developing systems based on multi-agents (MAS). The reason is that it can solve problems that an individual agent would have it difficult or impossible to solve [Mul]. Designing distributed software systems is pretty hard. At the same time, humans are very good at distributing work among them.

Such human organizations have inspired designing and using software agents, making software designs fit closer to our collective experience, providing a boost in stability, scalability, and maintainability of the software system [Xie17]. But, on the other hand, multi-agent systems are difficult to implement, because they require abstract modeling and higher level reasoning.

One of the platforms that implement MAS is Eve. Eve is a multipurpose, web-based agent platform. Agents on this platform can live and act anywhere: in the cloud, on desktops, in browsers, robots and others. The main protocol used in communication between agents is JSON-RPC [Eve].

Eve offers agents that are simple to develop; making this way eases the integration of software agents.

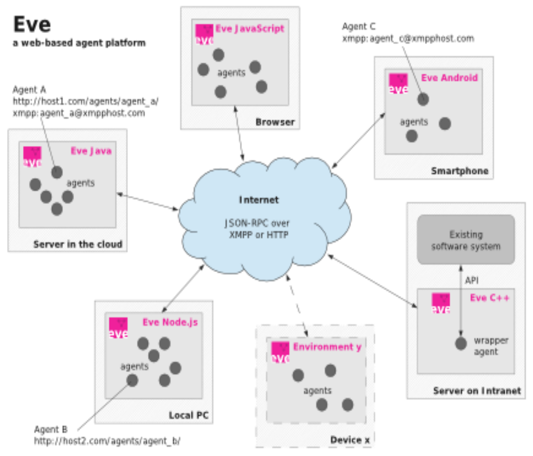

Figure 1: Eve platform

2 Traditional and Eve's approach

There are many multi-agent platforms, but Eve's approach to MAS makes it different from the other ones.

2.1 Traditional Approach

In traditional platforms, agents live and interact in an operating system or simulation environment [Bos10]. This is a closed and controlled environment, in the sense that its agents are deployed on a single server.

In order for the system to become scalable, the platforms must link multiple sites or locations together, enabling communications and interaction between agents that are on different sites. These platforms also provide the migration of agents from one site to another.

2.2 Eve's approach

We can say that Eve's approach for multi-agents differs a lot from the traditional ones. The core of Eve is to make agents available to other agents and to provide a way to help them communicate with each other.
Since agents can live on any device (servers, clouds, smartphones), Eve is platform independent.

In contrast to traditional platforms, whose agents live in a closed environment, the purpose of Eve is to offer agents an open, human-like world. Eve's agents live on the World Wide Web; they are accessed via a unique *url* and have a prescribed communication protocol.

Eve agents can be developed even on an existing application, because Eve can be hosted in any environment and can be written in any programming language [Ste14]. For this reason, applications do not have to adapt to fit into Eve; instead a simple layer needs to be added on top of the existing application in order to connect it to Eve.

Some advantages of the architecture of Eve that are important to mention are:

Scalability: Being fully web-based, new agents can be added to the system without lowering the performance.

Robustness: Eve is insensitive to the server or device failures, because Eve itself is a distributed agent platform, with no point of failure.

Massive parallelism of the workload of an agent: multiple instances of an agent sharing the same state can exist at once. This means that Eve allocates a limitless number of threads when an agent is heavily loaded, and no thread at all when the agent is idle. This reduces resource consumption for idle agents, in contrast to traditional platforms that allocate one thread per agent, even if the agent is idle.

Seamless migration: Since agents are fully location agnostic, accessing local or remote agents does not make any difference [Jon14].

3 Agent Modal

The main concept of Eve platform is an Agent, whose basic definition is: A software entity that acts on behalf of others in an autonomous fashion performs its actions on some level of proactivity and reactivity [Mah17].

The agents have an autonomous behavior, meaning that they need to run in an independent way of the entity they represents. In order to have this autonomous behavior, agents should have some features, which Eve provides:

Time independence: scheduling independent of the represented entity

Memory: the possibility to keep a model

Communication: a common language to communicate between agents [Eve].

3.1 Communication Protocol

The communication between Eve agents is achieved by the JSON-RPC protocol. JSON format is easily written by humans and easily generated or parsed by machines. Requests and responses sent by an agent that use this format. An example of a request is shown below:

Url http://myserver.com/agents/agent_y

Request

```
{
    "jsonrpc":"2.0",
    "id": 1,
    "method": "add",
    "params": {
        "a": 2.2,
        "b": 4.5
    }
}
```

Response

```
{
    "jsonrpc":"2.0",
    "id": 1,
    "result": 6.7,
    "error": null
}
```

Figure 2: Simple request from an agent and the corresponding response from another one

3.2 Agent methods

An Eve agent comes with the following methods:
-Agent.send(to: string, message: string) - Sends a message to the specified agent.
-Agent.extend(module: string | Array.<string> [, options: Object]) - extends the agents with one or more of agent modules listed in the next chapter.
-Agent.receive(from: string, message: string) - receives a message from the specified agent.
-Agent.connect(transport: Transport | Transport[] | string | string[], [, id: string]) - Connects an agent to one or more transports listed below.
-Agent.disconnect([transport: Transport | Transport[] | string | string[]]) - Disconnects the agent from the transport/transports specified. If no transport is provides, the agent will be disconnected from all the transports available [Api].

4 Eve concepts

4.1 Configuration

Local agents can communicate with each other using the default *ServiceManager*, loaded at *eve.system*. A System Manager is an object that manages services for agents and allows them to connect to relevant services [Mod].
A Service Manager configuration looks like below:

```
eve.system.init({
  transports: [
    {
      type: 'distribus',
    }
  ],
  timer: {
    paced: false
  }
});
```

4.2 Transports

Eve Platform has the following built-in transports:

Advanced Message Queuing Protocol (AMQP) is a protocol used for message-oriented middleware. Main features of this protocol include: message orientation, queuing, routing, reliability and security.

Distribus is a distributed message bus for node.js and the browsers. In this transport type, hosts are connected to each other in a peer to peer network. Peers then send messages to each other by their id.

HTTP is the easiest way of sending messages between agents. Messages sent by an agent are received by a web server and then sent back to the recipient.

PubNub provides publish/subscribe of a network.

WebSocket opens a connection between two agents. It is fast but its limitation is that it can be used only for a small number of agents, because of a certain amount of WebSockets that can be opened at the same time. Distribus is recommended when having to deal with large number of agents.

WebSockets can be used both server and client side. When a url is provided, WebSocket is considered as a server, where agents can connect. When url is not provided, WebSocket is a client that can be connected to other servers [Tra].

4.3 Modules

Eve Platform offers the following modules that agents can be extended with:

Babble - A Babble communication is modeled as a control flow diagram containing blocks: ask, tell, listen, if, decide, then. Blocks can be linked to the next block in the control flow and the blocks can dynamically determine the next block.

Pattern - The agents in this module will be extended with functions listen and do not listen. Having a certain pattern that can be a string or regular expression, a listener is triggered when an incoming message matches this pattern.

Request - This module offers the ability to send requests using the 'request' method and waiting for a reply.

RPC - Agents in this module to communicate using a JSON RPC protocol. This module can be used with all the transport ways listed above. Its 'request' method sends a request in JSON format and waits for its reply in the same format [Mod].

5 Use

The main methods of an agent are connected, disconnect, that are used to connect an agent to one or more transport types; send and receive, that are used for communication between agents.

Below are the steps that have to be followed in order to set up a system with eve agents. Create an agent class extending eve.Agent.

```
var eve = require('evejs');
 function MyAgent(id) {
   eve.Agent.call(this, id);
   // ...
 }
 MyAgent.prototype =
Object.create(eve.Agent.prototype);
 MyAgent.prototype.constructor = MyAgent;

 MyAgent.prototype.receive = function (from,
message) {
   // ...
 };
 module.exports = MyAgent;
```

Make necessary configurations, initialize transports and other services.

```
eve.system.load({
   transports: [
     {
       type: 'distribus'
     }
   ]
 });
```

Create an agent: *agent1 = new Agent('agent1')* and connect it to transports: *agent1.connect(eve.system.transports.getAll())*.

Send or receive messages using 'send' and 'respond' methods. The agent that sends or responds the request can be specified by its full *url* or just by its id:
agent1.send('distribus://networkId/agent2', message) or agent1.send('agent2', message) [Tra].

6 Conclusion

In this paper, we had an overview of the Eve platform, used for developing or integrating multi-agent systems to existing applications. The main advantage of this platform over the others was that Eve agents live in an open, human-like world and can be anywhere, in any device. Eve platform can be developed in any programming language, giving it another important advantage. We also saw the basic steps of implementing an agent-based system.

References

[Ste14] L. Stellingwerff, J. D. Jong and G. E. Pazienza. *Practical Applications of the Web-Based Agent Platform 'Eve'*, 2014.

[Mah17] Q. H.Mah17moud. *Software Agents: Characteristics and Classification, 2017.*

[Xie17] J. Xie, Chen-Ching Liu *Multi-agent systems and their applications*, 2017.

[Jon14] J. D. Jong and L. Stellingwerff.. *Eve: a Novel Open-source Web-based Agent Platform, 2014*

[Bos10] R. Bosch and D. Srinivasan. *An Introduction to Multi-Agent Systems, 2010*

[Eve] https://eve.almende.com/index.html

[Mul] https://en.wikipedia.org/wiki/Multi-agent_system

[Emsl] https://github.com/enmasseio/evejs

[Api] https://github.com/enmasseio/evejs/blob/master/docs/api.md

[Tra] https://github.com/enmasseio/evejs/blob/master/docs/transports.md

[Mod] https://github.com/enmasseio/evejs/blob/master/docs/modules.md