Interoperability of open-source VoIP and multi-agent systems

Marko Skomeršić[1], Neven Parat[2]
[1]Voice services department, Iskon Internet d.d., Garićgradska 18, 10000 Zagreb, Croatia
marko.skomersic@iskon.hr
[2]Elektrokem d.o.o., Augusta Šenoe 69, 10360 Sesvete, Croatia
nparat @elektrokem.hr

Abstract – In this paper the interoperability capabilities of open-source based VoIP (Voice over Internet Protocol) and multi-agent system will be explored and discussed. It will be shown how to successfully connect software agents with IP PBX (IP Private Branch Exchange) and how to make successful IP PBX monitoring using software agents. Applications and platforms based on open source, such as Asterisk IP PBX and JADE (Java Agent Development Framework) multi-agent platform, will be used in all segments of the system. This work is the first step in order to create fully automated VoIP system monitoring platform based on open source technology which is our final goal. As a proof of concept we will create and explain a simple multi-agent system that collects monitoring data from three open-source IP PBXs.

Keywords - VoIP, Asterisk, IP PBX, JADE agent platform, open source, IP PBX monitoring

I. INTRODUCTION

Growing popularity of open source based products such as Asterisk open source IP PBX [1] is a great motivation to use them in a various VoIP projects.

Open source VoIP solutions are used because of its numerous advantages compared to proprietary solutions. For example there is more control of entire system, wider span of experts who can perform maintenance and integration, there is no need for licensing fees, and it’s easier to customize the source code [2]. One of main advantages of open source - availability of source code free of charge to all parties that are interested to use it for any purpose - can also be exploited in finding security issues by malicious hackers. Most of open source projects don't have some commercial company or funding behind it which does not necessarily influence quality of software, but can prolong development, as it is mostly driven by enthusiasm and free time of programmers. Today, there is common opinion that open source products are free. This is not quite right. The code is free of any charge (with or without some kind of obligation to the developers depending on the distribution license), but there are fees for services like integration, implementation, customization and sometimes even for documenting the code and the product. So, initial investment should be quite smaller, but in a long time the overall price can be even greater than the proprietary software products. Nowadays there are many companies that are oriented to the open source market and that number is constantly growing. Digium Inc. is one example of open source based company. Digium Inc. is initial creator of the open source IP PBX called Asterisk.

Asterisk based VoIP system offers both classical PBX functionality and advanced VoIP features. It supports many communication protocols such as SIP (Session Initiation Protocol), H.323, MGCP, SCCP, E1/T1 PRI/BRI, IAX (Inter Asterisk Exchange protocol), etc. Asterisk IP PBX uses a modular software architecture which enables use of only needed modules. The architecture of dial plan, configurations and other Asterisk concepts will not be discussed in this paper. That can be found on the web.

The one of the many benefits of the open source products is its interoperability with other software products but there are some issues related to the security, reliability and availability. In such complex and immense VoIP systems there is a solution that can reduce or completely remove the influence of those issues. If we manage to set up efficient and constant monitoring system, we will be able to react in time to prevent potential damage.

In comparison with traditional client-server approach there are three main benefits of AOP (Agent Oriented Programming) approach. The first one is that mobile agents solve client/server network bandwidth problem. By moving a query or transaction from the client to the server, the repetitive request/response handshake is eliminated. Second, agents reduce design risk by permitting decisions about the location of code (client vs. server) to be pushed toward the end of the development effort, when more is known about how the application will perform. And third, agent architecture solve the problems created by intermittent or unreliable network connections, since agents can be built to work "off-line" and communicate their results back when the application is "on-line" [3]. So, the chosen approach is to use software agents as an automatic monitoring entity because of their main characterizations (autonomy, proactivity and an ability to communicate).

AOP is a relatively new software paradigm that brings concepts from the theories of artificial intelligence into the mainstream realm of distributed systems. AOP essentially models an application as a collection of components called agents that are characterized by, among other things, autonomy, proactivity and an ability to communicate [4]. There are few software agent platforms which match our main criteria that the system must be open source based. Because of its popularity the JADE platform [5] is used.

JADE is a software framework fully implemented in JAVA [6] programming language under GNU LGPL software license [7]. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA (Foundation for Intelligent Physical Agents) specifications [8].

This work will show how we can successfully integrate the software agents and IP PBX in order to collect some data from IP PBX and process it. This is only the first step to our final goal which is to improve reliability and availability of open source IP PBX using a multi agent system**.**

The problem was split into few main sub problems, each explained in its own chapter. The first sub problem is how to successfully create software agents using JADE multi-agent platform which will be able to communicate with Asterisk IP PBX. This is described in the second chapter (multi-agent architecture). Second sub problem is how to communicate with Asterisk IP PBX and which data we need to collect to make relevant conclusions about IP PBX load state. This is described in third chapter (interaction with Asterisk IP PBX).

Fourth chapter (security issues) will introduce reader with basic security issues in multiagent systems, and finally, fifth chapter (conclusion and further work) describes the potential benefits of this kind of architecture and related further work.

II. SYSTEM ARCHITECTURE

The system consists of two main subsystems. First is the access IP PBX which is physically placed into the network *DMZ* (DeMilitarized Zone - physical or logical subnetwork that contains an organization's external services to a larger, untrusted network, usually the Internet in order to add an additional layer of security to an organization’s LAN). It is used for VoIP communication with users outside the internal VoIP network. There are some remote users that are registered with the access IP PBX, but mainly it is used for trunking purposes with other IP PBXs outside internal VoIP network.

Second subsystem is internal VoIP network which consists of one IP PBX, media gateway, monitoring server and IP phones.

Booth IP PBXs are connected to the media gateway so they can call to and receive calls from the existing SOHO (Small Office/Home Office) network managed by preinstalled Ericsson MD 110 PBX which is connected to the PSTN (Public Switched Telephone Network). The media gateway is needed for converting voice media provided in one type of network to the format required for another type of network (i.e. SIP VoIP network to the ISDN). The complete architecture is shown in figure 1.

Asterisk architecture

Asterisk’s architecture is designed in a modular way in order to enable maximum flexibility. Specific APIs are defined around a central PBX core system which handles the internal interconnection of the PBX, cleanly abstracted from the specific protocols, codecs and hardware interfaces from the telephony applications. This allows Asterisk to use any suitable hardware and technology available now or in the future to perform essential functions, connecting hardware and applications. The complete Asterisk IP PBX software architecture is shown in figure 2.

The essence of Asterisk is PBX Switching system used for connecting calls between various users and automated tasks.

Application launcher is used to launch applications which perform services for users (i.e. voicemail, file playback, etc.).

Codec translator uses codec modules for the encoding and decoding of various audio compression formats used in the telephony industry (i.e. G.711a/μ, G.723, G.729, etc.).

Scheduler and I/O manager handles low-level task scheduling and system management for optimal performance under all load conditions.

There are four loadable module APIs (Application Programming Interfaces), facilitating hardware and protocol abstraction. Using this loadable module system, the Asterisk core does not have to “worry” about details of how a called is connecting, what codecs are in use, etc.

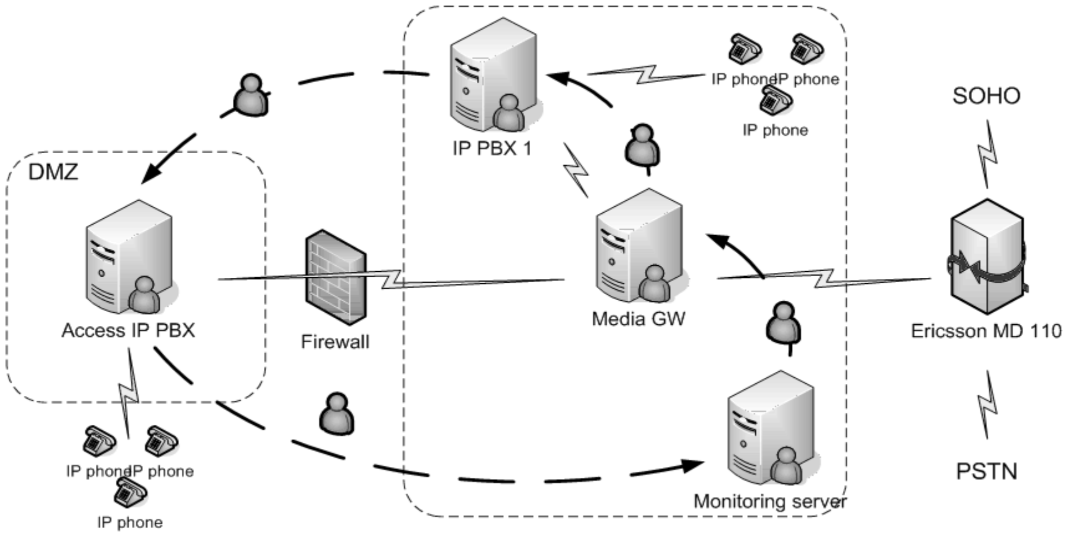

Fig. 1. Proposed architecture

Channel API handles the type of connection a caller is arriving on, it can handle VoIP connections, ISDN, Robbed bit signaling, etc. Dynamic modules are loaded to handle the lower layer details of these connections.

Application API allows for various task modules to be run in order to perform various functions.

Codec translator API loads codec modules to support various audio encoding and decoding formats such as GSM, G.711a/μ, G.729, etc.

File format API handles the reading and writing of various file formats for the storage of data in the file system. There are, also, ODBC (Open DataBase Connectivity) drivers for interaction with databases.

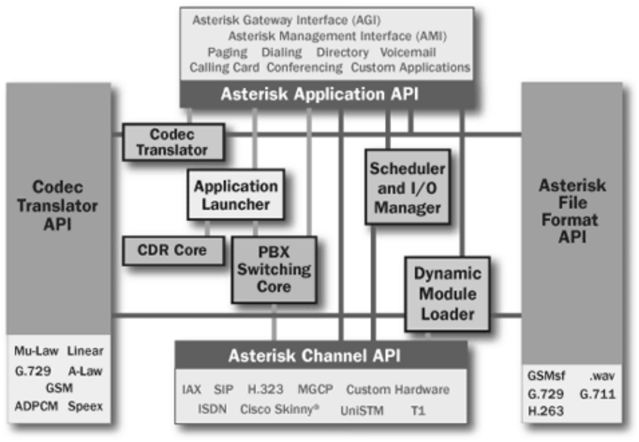

Fig. 2. Asterisk IP PBX software architecture

For communication with Asterisk IP PBX we will use Asterisk Management Interface (AMI). The Asterisk management interface API allows an external application to query and change Asterisk IP PBX state by sending actions and listening to responses and events. Basically it is a simple telnet application which connects to predefined AMI port.

II. MULTI-AGENT ARCHITECTURE

Multi-agent system consists of one distributed agent platform (JADE multi-agent platform) and three distributed containers within main agent platform. For every IP PBX in the system there is one remote container and there is one static agent (AsteriskAgent) in each container as shown in figure 1. AsteriskAgent communicate with Asterisk IP PBX trough Asterisk Management Interface (AMI) as shown in figure 3. in order to collect data regarding the IP PBXs current state. There is one mobile software agent (ColectorAgent) residing in a main container on a monitor server which communicates with all other static agents (AsteriskAgent) residing on remote containers in order to collect various informations such as current IP PBX load, number of registered SIP users, number of channels in use, etc.

After one complete cycle, CollectorAgent brings all collected data to the monitoring server which process collected data and present it to the administrator. CollectorAgent is constantly running its task until it is stopped by the administrator. This is shown on CollectorAgent's mobility diagram in figure 4. All of this collected data can be used to create statistics for future load projections, maintaince, etc.

Each software agent implements some kind of behaviors.

The AsteriskAgent has three behaviors. One for connection with AMI (*LoginBehavior*) which is basically a telnet connection setup on an Asterisk manager port (by default 5038), second behavior which collects monitoring data from Asterisk and third behavior, used for communication with CollectorAgent.

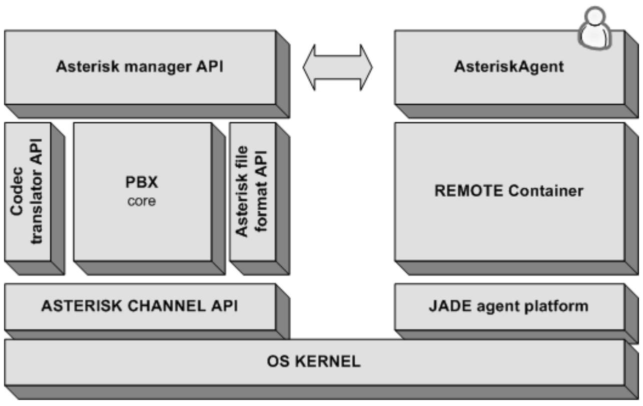

Fig. 3. Communication between AMI and a static software agent (AsteriskAgent)

The CollectorAgent has, also, three behaviors. One for discovering registered AsteriskAgents and their addresses, second for communication with AsteriskAgent and third for delivering collected data.

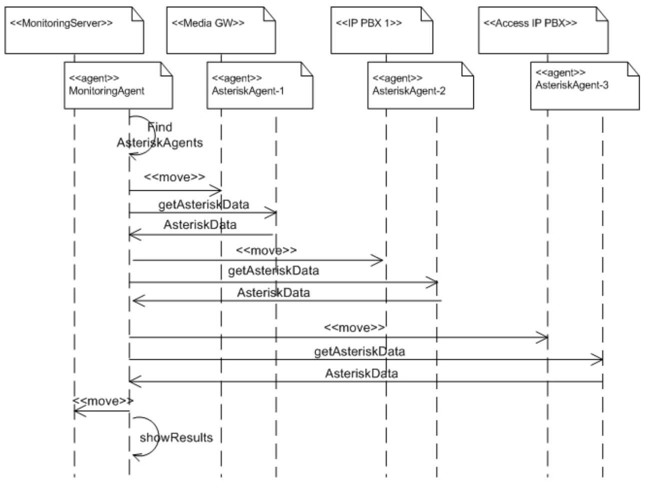

Fig. 4. CommunicatorAgent cycle mobility diagram

III. INTERACTION WITH ASTERISK IP PBX

As previously said, AsteriskAgent is used to communicate with Asterisk IP PBX trough AMI using asterisk-java package [9] in order to collect relevant data. When AsteriskAgent starts it registers its services in yellow pages (DF – Directory facilitator) so it can be found by CollectorAgent. The source code for registration of service is shown in fig. 4.

```
DFAgentDescription
      dfd=newDFAgentDescription();
dfd.setName(getAID());
ServiceDescription
      sd=new ServiceDescription();
sd.setType("asterisk-show-channels");
sd.setName(getLocalName()+
      "-asterisk-action");
dfd.addServices(sd);
```

Fig. 4. Source code for agent service registration

LoginBehavior is used to log onto Asterisk IP PBX via AMI. It is started when AsteriskAgent starts. Used parameters are: IP address/domain name and port of the IP PBX server, AMI username and AMI password. AMI must be configured to accept that connection request. This configuration is in flat file on IP PBX file system called *manager.conf* as shown in fig.5.

```
[HomeAgent]
secret = Marko123
deny = 0.0.0.0/0.0.0.0
permit = 10.2.1.0/255.255.255.0
displayconnects = yes
```

Fig. 5. Configuration of *manager.conf* file

Second behavior, called *ActionBehavior*, is used to invoke some action on Asterisk IP PBX. These actions can be all available Asterisk CLI (Command Line Interface) actions [10] passed to the asterisk as a string parameter.

In our proof of concept we invoke *SHOW CHANNELS* action. When invoked, *SHOW CHANNELS* action displays the total number of currently used voice channels and its type. The type of channel can be: Zap channel mainly used for ISDN interconnections, IAX2 channel which is Asterisk's trunk channel type, SIP, H.323, etc. *SHOW CHANNELS* action also displays a total number of active calls. Complete output of *SHOW CHANNELS* action is shown in figure 6.

```
Channel Location State   Application(Data)
SIP/300 (None)   Up   BridgedCall(SIP/312)
SIP/312 macro-stdexten Up Dial(SIP/300|20)
SIP/300 (None)   Up   BridgedCall(SIP/313)
SIP/313 macro-stdexten Up Dial(SIP/300|20)
4 active channels
2 active calls
```

Fig. 6. *SHOW CHANNELS* action console output

The received set of informations is parsed in order to collect only the number of currently active calls. This number is stored to pass it to the CollectorAgent. AsteriskAgent can be modified to collect data from IP PBX in some time interval (i.e. each two seconds) so statistics can be passed to the CollectorAgent.

For our proof of concept we didn't implement any GUI (Graphical User Interface) on a collector agent so it displays collected data on a console output as shown in a figure 7.

```
IP PBXs found: 3
IP PBX 1: <IP ADDRESS 1> ; 0 active calls
IP PBX 2: <IP ADDRESS 2> ; 2 active calls
IP PBX 3: <IP ADDRESS 3> ; 1 active calls
```

Fig. 7. CollectorAgent console output

IV. SECURITY ISSUES

Traditionally security threats such as masquerading, eavesdropping, spoofing, service misuse, denial of service, and tampering of data or manipulation of data are also applicable for FIPA-based agent systems. These security threats relate to the confidentiality, integrity or the availability of the agents and should be considered when developing an agent system [11].

In our, proof of concept, work there is none of security techniques implemented and this has been left for further research and implementation. We propose some solutions which will greatly improve overall security of proposed system. Some of this techniques are using PKI (Public Key Infrastructure) as discussed in [12].

V. CONCLUSION AND FURTHER WORK

This work shows that it is possible to connect open-source IP PBX with new paradigm of agent oriented architecture in order to, successfully, create advanced communication system.

Of course, as all things in a real world, agent based approach has its strengths and limitations. Main advantages of AOP are reduced frequency of network use (bandwidth requirements and repeated interactions), increased asynchrony between clients and servers so there is no need for long reliable network connections, increased distribution and reconfiguration of services which manifests in overload avoidance and ease adoption to individual requirements. AOP increases concurrency in the system which enables task decomposition among multiple agents so parallel activities can be accomplished. Some main disadvantages regarding AOP are security issues consisting of identification, authentication, and protection from viruses or malicious agents. There is also issue with transport and migration demands which increase software complexity [13].

These two different worlds can be applied in various applications. We can use agents to monitor call load on each IP PBX and based on results to transfer some calls to the PBXs which are not overloaded, we can use it to improve availability of the whole system which is ore final goal. In order to achieve this we need to setup some virtual IP PBXs on physically different servers so agents can power them up or down if some unexpected scenario occurs (i.e. one IP PBX fails on server A, agent can power up the same virtual IP PBX on server B).

This is our first work on specific topic, in our further research we will try to setup a complex system with prepared virtual IP PBXs (cloned from real ones), central storage, call detail record, database and registrar server in order to research some high availability scenarios and calculate improvement of availability if any.

REFERENCES

[1] Asterisk an open source IP PBX, http://www.asterisk.org, Digium Inc., 2007.

[2] G. Camarillo, „SIP Demystified“, The McGraw-Hill Companies, Inc. 2002.

[3] T. Sundsted, “Agents on the move”, JavaWorld.com, 1998.

[4] F. Bellifemine, G. Caire, D. Greenwood, „Developing multi agent systems with JADE“, John Wiley & Sons, Ltd, 2007.

[5] F. Bellifemine, G. Caire, A. Poggi, G. Rimassa, „JADE a white paper“, http://jade.tilab.com, October 2007.

[6] JAVA programming language, http://java.sun.com/, November 2007.

[7] GNU LESSER GENERAL PUBLIC LICENSE, version 3, http://www.gnu.org/licenses/lgpl.html

[8] FIPA web site, http://www.fipa.org

[9] Asterisk-java official web site, http://asterisk-java.org/, November 2007.

[10] Asterisk CLI web site, http://www.voip-info.org/wiki-Asterisk+CLI, December, 2007.

[11] Siv Hilde Houmb, “Security issues in FIPA agents”, NTNU, 2002.

[12] Hu, Y.-J., Some thoughts on agent trust and delegation, Proceedings of the fifth international conference on autonomous agents, pages 489-496, ACM Press, ISBN 1-58113-326-X, 2001.

[13] Ignac Lovrek, “Soft mobility, Part I: Mobile software agents”, FER Zagreb, Department of telecommunications, February 1999.