Revista Venezolana de Computación
ISSN: 2244-7040
http://www.svc.net.ve/revecom
Vol. 5, No. 1, pp. 20-28, Junio 2018
Fecha recepción: 07/03/2018, Fecha aceptación: 02/08/2018

Integrating a Unified Communications System with Social Networks

Eric Gamess[1], Francisco Mora[2], Diego Oliveros[2], Dedaniel Urribarri[2]
egamess@jsu.edu, moraf93@gmail.com, oliverosdiego17@gmail.com, dedanielu@gmail.com

[1] MCIS Department, Jacksonville State University, Jacksonville, AL, USA
[2] School of Computing, Central University of Venezuela, Caracas, Venezuela

Abstract: The increasing number of employees and clients, that are associated with each organization, has motivated them to extend their technological platform with the implementation of unified communications servers, with the goal of significantly improve their communication processes. On the one hand, people usually look for easy and fast methods for their communications. On the other hand, social networks have experienced an exponential boom in the last few years. Hence, in this research work, we propose a solution that allows the integration of a unified communications system with a social network. For the unified communications system, we choose Elastix, since it has become very popular and many companies worldwide have already based they communications system on it, whereas Twitter has been our selection for the social network, since it is commonly used for the exchange of short and accurate information among people. Our proposal is focused on the customer services offered by organizations to people, using Twitter.

Keywords: Customer Services; Unified Communications Systems; Elastix; Social Networks; Twitter; Integration.

I. Introduction

Nowadays, communication technologies are becoming so important that they form part of the strategic plan of organizations. In the actual globalized world, people take an important part of their workday to see emails, make calls, use instant messaging, and even participate in video-conferences to communicate with collaborators or customers. Therefore, communications systems have a great challenge ahead, which consists of offering the required tools to organizations with the aim of improving their way of doing business.

IP telephony is a technology that is implemented on top of the existing data networks. This technology has been in the market since the late nineties, but has not been widespread until recently, thanks to the improvement and standardization of the systems that provide voice quality control and the universalization of the Internet service. IP telephony usually brings an efficient and flexible environment in an organization for communications, and allows remote locations to be smoothly integrated into the headquarter.

On the other hand, social networks allow a fast, effective and simple interaction among a large number of people. The focus of the organizations in the usage of social networks cannot be limited to the advertisement of products and promotions, but must also point to the resolution of doubts and problems of customers, and to maintain this direct contact through the Internet.

In this work, we propose a solution to integrate or extend the functionalities of unified communications servers with social networks.

The rest of this document is organized as follows. In Section II, we present the problems faced by organizations with their communications platforms. In Section III, we introduce the different elements that were considered as possible part of a software solution. Related works are reviewed in Section IV. In Section V, we present our proposed software solution for the integration of unified communications systems with a social network. Section VI describes the scenarios used and the tests performed to validate our solution, while the results are discussed in Section VII. Finally, Section VIII concludes the paper and gives directions for future works.

II. Problems Faced by Organizations with their Communications Systems

As time goes by, the growing limitations of the old telephone system can be evidenced. For example, the conventional telephone system presents serious problems of scalability, especially when it comes to add new telephone lines. In addition, it also has low flexibility when the users want to develop and implement specific applications that meet the needs of a particular company. For these reasons, worldwide, we can observe that entities are migrating their telephone systems to VoIP.

In the last few years, usage of technology has increased significantly in all areas; in particular information technologies are now present in all processes carried out in our society. For example, social networks are now ubiquitous, leading most people to change both, their personal and professional behaviors.

However, even though social networks are becoming widespread, they are still rare in unified communications services, particularly in the following areas: process management, customer service, marketing, and advertising. These areas can be significantly supported and improved by integrating unified communications services with social networks. Currently, the integration between state-of-the-art telephone systems and social networks are practically non-existent; therefore, there is no work and substantial evidence of the benefits that social networks would offer to unified communications systems, by expanding the services currently available.

III. Study of the Possible Technologies

During the last decades, the experience offered to customers by call centers has evolved. This is due to the fact that the communication with the clients is not anymore only focused on incoming and outgoing voice calls, but now also integrates data applications like e-mail, web-based chat, instant messaging, and the capability to share pictures and web pages sent to and from the customers. Hence, the term "Call Center" is now becoming obsolete, and the one used nowadays is "Contact Center," with the encompassment of all communication channels with clients.

In our society, that has been flooded by social networks, it is logical to think that these channels of communications can be used not only for the interactions between individuals, but also to communicate customers with large companies and organizations. Through social networks, a user can request information about a product, generate a claim, express an opinion, or require services. In this research work, we propose the integration of two important technologies: (1) Elastix as the unified communications server and (2) Twitter for the social network.

A. Elastix

Elastix [1][2][3][4] is an open source software platform that aims to incorporate in a single solution all media and communication alternatives available in the business world [5]. Its functionality is based on the usage of four very important software programs: (1) Asterisk [6][7][8][9], (2) HylaFAX, (3) Openfire [10], and (4) Postfix [11][12][13]. These software provide functions of PBX (Private Branch Exchange), fax, instant messaging, and email, respectively.

1) Architecture: Elastix is not only in charge of providing telephony, but it also integrates other means of communications to make the work environment efficient and productive. By having an integration of different communication systems, an enhanced productivity is achieved in different aspects such as saving time and paper, facilitating the access to shared information, among others. Figure 1 shows the different layers of communications present in the general architecture of Elastix.

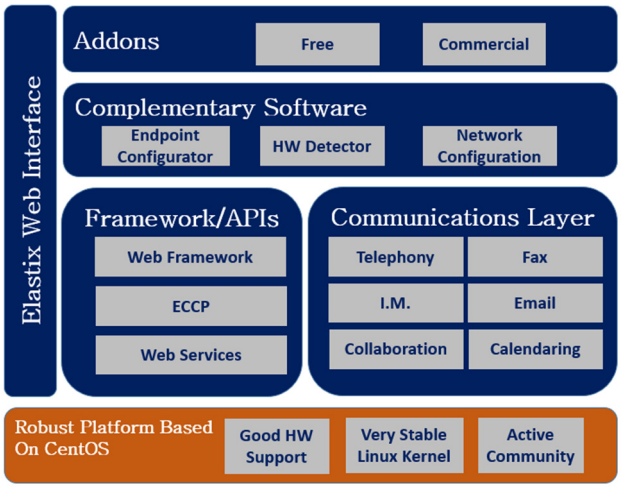

Figure 1: Architecture of Elastix

2) Characteristics: Elastix has multiple features and functionalities related to the services it provides: IP telephony, fax server, mail server, conferences, instant messaging, among others. These characteristics are provided by the software on which Elastix is based, mentioned above. The latest versions of Elastix allow third parties to develop software modules to improve the system, or to customize it for their own use. Below, we describe the technologies that help developing additional features to integrate them into Elastix.

Elastix Call Center Protocol (ECCP)

It is a text protocol based on XML and specialized for call centers. Its objective is to provide client applications with a single comprehensive protocol as an alternative to other existing protocols such as web services or AMI (Asterisk Manager Interface).

One of the disadvantages of this protocol was its performance, since initially, the only scheme available was based on the concept of "polling," which generated numerous queries to the server and an unnecessary waste of Elastix server resources. The new version of the protocol supports asynchronous events for communications, eliminating the need for "polling" and offering the possibility of a scalable solution.

This protocol provides a communications API (Application Programming Interface) available through a TCP port to which client applications can connect in order to communicate with the predictive dialer of Elastix, allowing third parties to develop their own agent consoles or other types of client applications.

Asterisk Gateway Interface (AGI)

This interface [14][15] is mainly used to add new features to Asterisk through the use of different programming languages, such as Perl, PHP, C, Pascal, among others. Its function is to connect the Asterisk dialplan with an external program that seeks to manipulate a channel in the dialplan. The interface is synchronous, i.e., the action taken on a channel by an AGI block does not return until the action is completed.

Asterisk Manager Interface (AMI)

AMI provides a mechanism to control where the channels are executed in the dialplan. Unlike AGI, AMI is an asynchronous interface, by events. For the most part, AMI does not offer mechanisms to control the channels' execution; rather it provides information about the state of the channels and where the channels are running.

Both interfaces (AGI and AMI) are powerful and open a wide range of integration possibilities. AGI enables the execution of the remote dialplan, which allows developers to control the channels in Asterisk using PHP, Python, Java, and other languages. With AMI, the Asterisk status can be displayed on the screen, calls can be initiated, and controlled channels can be located. By using both APIs together, it is possible to create complex applications using Asterisk as the engine for the development.

Asterisk RESTful Interface (ARI)

ARI allows developers to build custom communications applications. ARI exposes primitives of Asterisk that are normally reserved to C modules (channels, bridges, endpoints, communications media, etc.) through an intuitive REST interface. ARI transmits the state of the objects that are controlled by the users, through JSON [16][17] (JavaScript Object Notation) events over a WebSocket [18][19].

By giving control of the fundamental building blocks in Asterisk to all developers, regardless of the programming language, Asterisk is becoming a communications engine, with the business logic of how things should be communicated, delegated to the application using Asterisk.

ARI is not a substitute for AGI or AMI. Rather, it is a complementary API:

- AGI allows users to control the execution of dialplan applications on remote processes.
- AMI allows users to manage and control calls at a high level.
- ARI allows the replacement of dialplan applications with user-customized communications applications.

B. Social Networks

Social networks have revolutionized the concepts of personal relationships and entertainment. They are aimed to maintain contact with people, create new links, interchange information and opinions, and furthermore, they can be used in many other areas, such as finding job opportunities.

Every day, millions of people around the world use social networks such as Facebook and Twitter to express their opinions. This represents a valuable opportunity for organizations to know more about their actual or potential clients, in addition of reaching a greater number of people, that could not be done in a traditional way.

1) Principal Use of Social Networks: Initially, social networks were intended to connect people, with a focus on personal communications. However, over the years, they became a place where organizations can interact with customers and propose their products and services. According to the classification carried out by Del Moral [20], the four main usages of social networks are:

- Maintenance of friendships: keeping in touch with friends, colleagues or ex-partners of work, summer acquaintances, etc. In the past, before social networks, many of these relationships will not last in the long run, due to the difficulty and cost of communications.
- New friendships: while social networks facilitate the maintenance of contact between people who know each other, they also promote new contacts between people. For example, in most of these social networks, users can define friends. The list of friends is generally visible to other contacts and friends, who can in turn interact and meet each other. Thus, the friend of a friend can become a contact and later a friend of a third party. This converges on the "six degrees of separation" theory from Frigyes Karinthy, which suggested that we would not need to contact more than six people to find someone, following their networks of friends and acquaintances. In other words, any two people on the planet are linked, without knowing it, by a chain of friends or acquaintances, with a length of at most six people.
- Entertainment: although social networks serve to interact and increase relationships, there is also a profile of users who use them as an entertainment portal. These users explore the updates of the state of other users, that is, they inform themselves about other people's lives, they discover the new colleagues of former classmates, etc. It is a way of observing what is happening without being seen.
- Group of related people: it is one of the main usages of social networks. People that share the same interest or from the same professional sector can group themselves to discuss their common interest. Furthermore, some organizations create private social networks to streamline procedures, communications, conferences, or reports.

2) Usage Focused to Organizations: For organizations, it is important not to merely used social networks as an advertising platform. They must be integrated to promote socialization, the exchange of experiences, etc. In fact, using social networks mainly as a sales platform can generate a negative perception of the brand, given the social, rather than commercial, expectation that the term "social network" implies.

There are many benefits that a correct strategy for the usage of social networks can generate, and these will depend not only on the strategy as such, but also on other external factors such as the sector of the organization, the commercial activity, the size of the organization, the target community, and last but not least, the level of commitment of the organization with the implementation of a digital marketing plan.

3) APIs for Social Networks: An API (Application Programming Interface) is a set of functions/methods that provide the programmer with an interface of communications with a specific system, allowing him/her to develop new custom functionalities. In our days, social networks have information from users that is quite useful to promote products

and services. Currently, many applications are being designed in such a way that they have the ability to establish a connection with the API of these networks, and obtain relevant data from the users, in order to customize the information to be shown to them.

To establish such a connection, a process of authentication and permission authorization must be followed, through the usage of the "OAuth" protocol [21][22][23] (Open Authentication). This protocol allows a user to grant access to his/her data to a third party, without having to provide his/her username and password. In this process, when the user grants permission to the application, the social network provides a "token" that must be saved by the application in order to make requests on behalf of the user, such as reading personal information, interests, contacts, or publishing new information.

The interaction between social networks and the application is made through requests, sent with the HTTPS (Hypertext Transfer Protocol Secure) protocol [24]. According to the action to realize, requests of type GET, POST, PUT or DELETE can be used. These requests are analogous to the actions of reading, writing, editing or deleting, respectively. All requests must include the access "token" through which the request is validated, accepting or rejecting according to the permissions granted to the user. It is recommended to send the access token within the header of the requests, although it can also be sent as a parameter in the URL, since it is protected by the HTTPS protocol.

Even though all social networks have similar APIs, they have different restrictions for the management of data and a different usage in society.

Twitter

It is one of the social networks that has done a lot of efforts to promote its API. In fact, the statistics indicate that more than half of the accesses to their tweets are made from external applications, such as TweetDeck [25] or Seesmic. There are thousands of products developed from the API of Twitter. Some of them, that require a very high volume of information, pay for it, resulting in a new business model.

Twitter is one of the social networks that does not have many privacy restrictions, that is, the vast majority of users have a public profile, and therefore also are their "tweets." For this reason, Twitter allows access to all this information through its API.

One of the restrictions that developers face is the limit of requests that can be done in a period of time. That is, there are 15-minute intervals where a maximum number of requests can be made. It is worth mentioning that these limits are per user, not per application, allowing an independent control on each user. According to the type of resource that is requested, there are two main types of restrictions: (1) 15 requests every 15 minutes and (2) 180 requests every 15 minutes. Additionally, it is not allowed to retrieve historical information, that is, if a search is executed, it is only possible to obtain information generated in the previous seven days. In case of exceeding the maximum number of requests in a period of time, a response code is obtained which provides information about the temporarily restricted resource and the waiting time for the resource to be available again. However, Twitter also offers the usage of streaming, which allows the acquisition of information in real time, without restricting the number of requests for a period of time.

IV. Related Works

A few works have been done in relation to the integration of unified communications systems with social networks. In this section, we describe some of these works.

A. Integrating Elastix with Gtalk

The work carried out by Gaibor [26] consisted of the integration and configuration of the Gtalk instant messaging service in Elastix (see Figure 2). The author discovered that the Asterisk version which comes with Elastix 2.2 brings a compiled Gtalk support, allowing an easy communication with the service.

In the configuration process, the file of the XMPP protocol (formerly known as Jabber) had to be modified to create a user, by using a Gmail account. Then, a context for the user had to be made with a file of custom extensions, allowing the execution of a series of commands when calling the extensions. Finally, it was added to the dialplan and tests were done to verify the correct operation of the user.

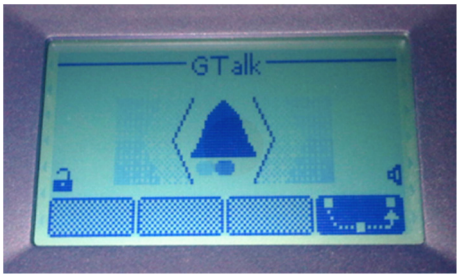

Figure 2: Call Received from a Google Account

B. Publication in Twitter through ASR

The work done by Smith [27] is about writing a tweet in a personal Twitter account, through a call to an extension and commenting the content of the tweet using an ASR (Automatic Speech Recognition), at no cost. Smith used PHP-AGI, a library to work with AGI from the PHP programming language, and wrote some scripts. The author also used a WAV (Waveform Audio Format) to FLAC [28] (Free Lossless Audio Codec) converter. WAV is a digital audio format that does not have data compression, while FLAC allows digital audio to be compressed without losing information. The conversion was required, since the solution uses the free ASR service [29] from Google that only receives FLAC formats. In order to convert from WAV to FLAC, Smith used the SoX [30] program, a famous tool to convert, add effects, and other advanced sound manipulation functions, on audio files, from a terminal.

Finally, Smith had to create and authorize an application in Twitter, so that the PHP script could work. The application required read, write, and access permissions to direct messages, in order to manipulate the account. The most important information that could be obtained from the application are: the consumer key, the consumer secret, the access token, and the access token secret.

V. Integrating Elastix with Twitter

Figure 3 depicts the conceptual design of the proposed solution, which is divided into two modules distributed in different physical servers. The elements involved in our solution are explained next.

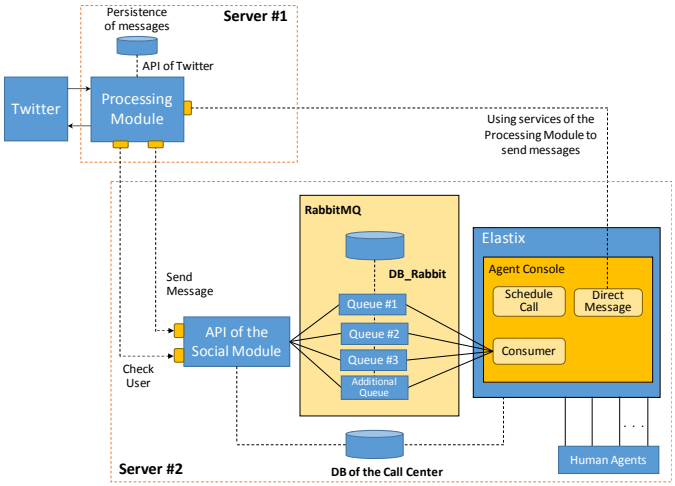

Figure 3: The Proposed Architecture to Integrate Social Media to Elastix

A. Twitter

It is a software component that acts as an intermediary between the organization and its clients. It receives the requests from the clients, sorts them, and publishes the associated responses. The interaction with the API of Twitter is based on having a Twitter account with permissions for third party apps, so the application can read and write direct messages. Direct messages is the tool that we selected to gather the important data used by the processing module (see Section V.B). Since the data exchanged can be sensible, we chose them since they offer an exclusive private bidirectional canal of communication with the owner of the account (client). In addition, unlike the tweets, direct messages do not have the limit of 140 characters in length, allowing the exchange of longer messages.

B. Processing Module

The processing module uses the services of the Twitter API to obtain and send direct messages to clients. Due to the asynchronous nature of the flows between Twitter and the social module (see Section V.C), it was necessary to label the direct messages with a tag since they can be in one of three possible states: (1) the direct message was satisfactory processed in the social module, (2) the direct message is been reviewed in the social module, and (3) the delivery of the direct message to the social module has failed.

This module was developed with the Rails framework [31] for the flexibilities and advantages it offers at the time of programming. We also used the REST architecture and the JSON [16][17] (JavaScript Object Notation) format for data transfer, due to their simplicity, their speed of processing, and since the Twitter API also manages these technologies. Figure 4 shows the different functionalities that reside in the processing module for the handling of messages from the social network.

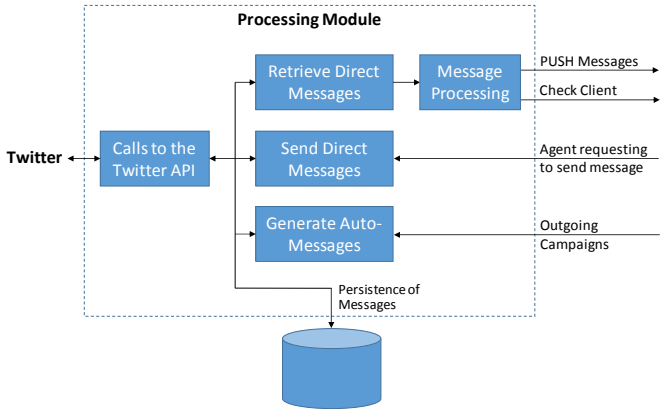

Figure 4: Processing Module - Elastix Social Media

1) Call to the Twitter API: This component is in charge of configuring the information of the account associated with Twitter, to make calls on behalf of the owner of the account. To do so, it uses a gem of Rails called "Twitter," and must obtain from the Twitter application some secret and unique values: consumer key, consumer secret, access token, and access token secret.

After the configuration and thanks to the used gem, it is possible to tweet, send and get direct messages through simple directives such as:

- create_direct_message(<ID_or_UserName>,<Message>): send a direct message without limit of characters to a user by specifying his/her ID (identification number) or username.
- direct_messages: retrieve all the direct messages of the configured account.
- update(<tweet>): update the timeline of the account by sending a new tweet (as previously specified, tweet are limited to 140 characters).

2) Retrieve Direct Messages and Message Processing: The function of these two components is to use the Twitter directives to obtain all direct messages and to perform the classification of the obtained messages. The classification is done as (1) complaint, (2) doubt, or (3) compliment, and messages have a specific structure depending on their classification.

The messages that are retrieved are stored in a database and the following labels are kept: waiting, delivered, and finished. These labels correspond to the receiving process of a message through the API, its submission to the social module for its attention, and the effective response of the agent to the user's request.

3) Send Direct Messages: It is a web service to be used by the agent for the submission of the answer to the request with a direct message to the user.

4) Generate Auto Messages: This component is used for automatic messages and outgoing campaigns. These messages

are classified as "Informative Tweets", "Outgoing Campaigns by Tweets," and "Outgoing Campaigns by Direct Messages".

C. *Social Module*

This component is in charge of the operations on the messages received by the processing module, in addition to providing support tools for the management of the contact centers in relation to their link with social networks. Its structure is part of the ecosystem of the unified communications server, where it maintains dependencies with the rest of its components. The "Social Module" in its operative scope maintains the management of the service queues and the distribution scheme of messages to the agents, allowing them to support their administration with the generation of calls, submission and reception of direct messages, management of outbound campaigns in social networks, among others. Figure 5 shows the components involved in the operation of the social module.

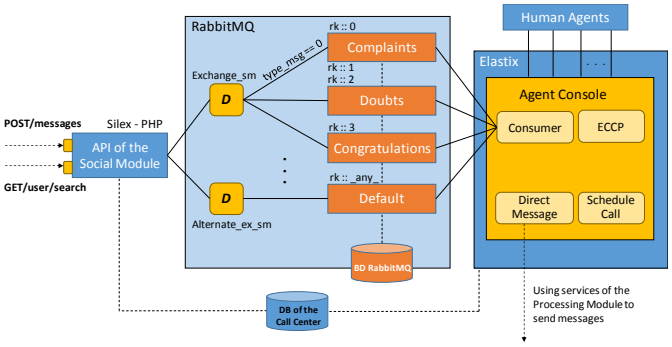

Figure 5: Social Module - Elastix Social Media

1) API of the Social Module: Given the approach proposed for the communication between modules, we implemented an API with Silex [32], a PHP micro-framework, to get an intermediary between the processing module and the service queues handled by the RabbitMQ message broker [33][34][35], which is responsible for providing the services required to complete the workflow of each message. In the following bullet list, two important REST web services are discussed:

- messages: service obtained through a POST request to enqueue in RabbitMQ a message previously processed and parsed in JSON format.
- /user/search: service obtained through a GET request, to verify the existence of a client within the module of the call center, by sending two of its identifiers (identification number and contract number).

2) Service Queues: The service queues managed by the social module are supported by additional structures to the existing ones in Asterisk/Elastix, given the peculiarity of the attributes that are stored in them, that are far from those handled in the telephony field. That is, they store messages processed by the processing module that will be consumed by the available agents. For this, they exploit the benefits of the RabbitMQ message broker, to make routing, persistency and handling of the messages that are delivered through the API of the social module.

Two routers are used to deliver the messages to four different queue structures that have configurations that enable specific persistence functionalities and work mode (ACK waiting). Each of these queues represents a different flow over the complete system, since each message has specific information, that is, each one receives specific data depending on the type of messages in which it was categorized by the processing module, according to the request generated by the client.

There is a one-to-one correspondence between the type of message and the routing key that each queue has inside RabbitMQ, which is an integer between 0 and 3, representing complaints, doubts by calls, doubts by direct messages, and compliments, respectively. Likewise, there is an additional queue managed by an alternate router, which was implemented for reasons of future expansion and to cover the loss of any user message that is routed with a different "key route," than the one known within the solution.

3) Call Center Module: The Elastix Call Center module is designed to handle incoming and outgoing call campaigns, allowing the interaction between agents and telephone service subscribers. For the purpose of integration with social networks, the messages found in the different queues of RabbitMQ that correspond to the incoming campaigns and the outgoing campaigns are used when scheduling a call. In this scheme, the customer is within the Elastix environment, specifically in the agent console, where a panel was developed where all the requests associated with Twitter and the users affiliated to the organization must appear.

4) Social Panel: Given the nature of the agent console to provide customer service through telephone calls, in the development of the social panel, different types of responses were enabled that agents can use to solve the requests obtained from the social network. In our case, the answers are handled through direct messages to users, or a call is planned to them. In the social panel, there is a button called "Enable Social Media" to enable the process of consumption of messages in RabbitMQ. When activated, a series of flows that are explained below are triggered:

1. Instantiation of a RabbitMQ consumer.
2. Acquisition of messages from the queues subscribed by the consumer.
3. Visualization of the information of the messages in the agent console (Social Panel).
4. Change of the displayed labels in certain buttons of the view.
5. Initialization of two timers for the attention of each instance of an active agent console.

Figure 6 depicts the agent console with the Social Panel activated.

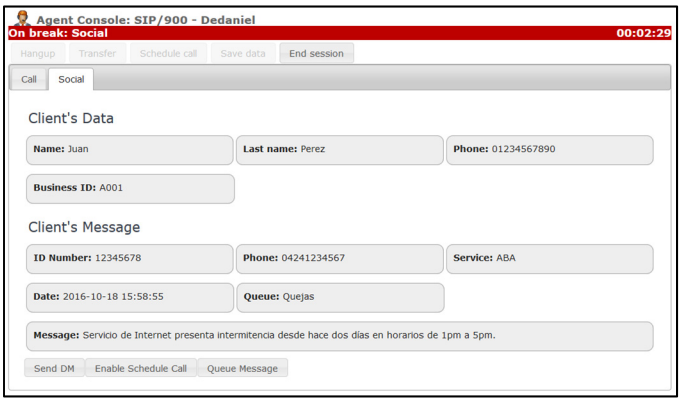

Figure 6: Agent Console – Social Panel

The operation mode of the social panel is cyclical. When the social mode is activated, a message is consumed from the service queues of RabbitMQ. The agent console takes care of it, and finally an intermediate time of action is given before the next message is consumed.

VI. Tests and Particular Situations

Table I describes the set of tests associated with performance and load, which were done to determine the impact of the social module on the Elastix communications server. In this way, we were looking to evaluate and determine if the implementation of this module is feasible on a production environment in terms of response times so that it can be used in a real customer service. To carry out the tests, we used PHPUnit, a well-known PHP framework for testing. With PHPUnit, users can adjust the values according to the scenario of each test, depending on the requirement to be evaluated.

Table I: Description of the Tests

ID	Name	Description
1	Processing Direct Messages	Determine the average time and the impact of processing direct messages obtained in the PM (Processing Module) from the Twitter API.
2	Submitting Direct Messages	Determine the average time and the behavior of the application when generating a direct message from the agent console.
3	Retrieving Direct Messages	Determine the average time and the impact when retrieving the direct messages sent to the associated account of the application.

VII. Performance Validation of our Proposal

A. *Time for Processing Direct Messages*

The purpose of this test is to determine the average time which elapses between the acquisition of a message from the processing module, until it is sent to its respective service queue.

For this experiment, we generated 50 messages from the social network to the different service queues: 20 associated with the complaint queue, 20 for the doubt queue, and 10 for the congratulation queue. Figure 7 depicts the results that we obtained. The x-axis represents the experiment number (from 1 to 50), while the y-axis is the elapsed time in milliseconds. Our experimental results show that the elapsed time is acceptable for most of the experiments, that is, with a value under 2000 ms. The picks are due to high network traffic and load between our testbed and the Twitter servers, which are related to the Internet services.

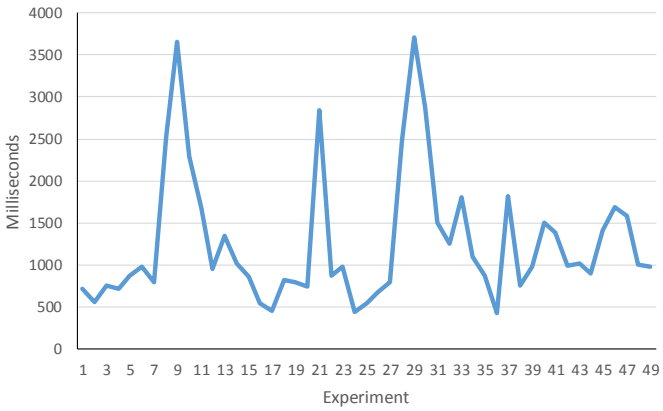

Figure 7: Response Times when Processing Direct Messages

B. *Time for Submitting Direct Messages*

The goal of this test was to evaluate the effective time to get and consume the web service of the processing module, which is responsible for sending direct messages to Twitter users, specifically those who have been answered from an instance of the agent console.

For this experiment, we generated 50 responses from the agent console to the 50 messages obtained from the service queues of the previous experiment. Figure 8 shows the results that we obtained. The x-axis represents the experiment number (from 1 to 50), while the y-axis is the response time in milliseconds. Our empirical results show a response time that is acceptable for most of the experiments, that is, with a value under 1500 ms. It is worth remembering that this time should be lower than in the previous experiment, since the consumption was directly against the processing module, and not against the service queues.

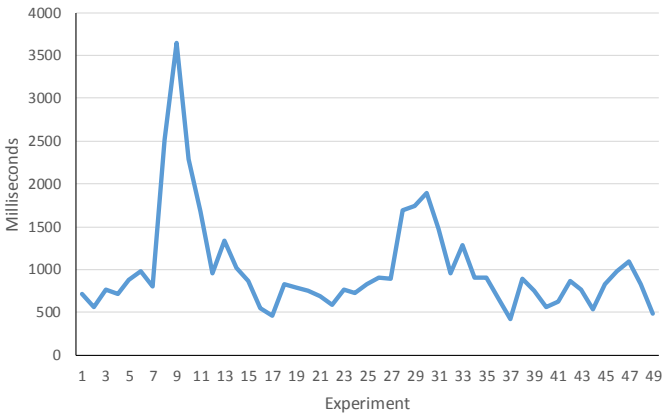

Figure 8: Response Times when Submitting Direct Messages

C. *Time for Retrieving Direct Messages*

The objective of this test was to determine the average time required by the Processing Module to retrieve all direct messages from the social network, that is, the time that elapses between a request sent by the Processing Module to Twitter to obtain all direct messages associated with a specific user account, and the reception of those. To do so, we generated 50

requests from the Processing Module to the social network (Twitter) to obtain the direct messages associated with the @SocialMedia_App account.

Figure 9 shows the results that we obtained. The x-axis represents the experiment number (from 1 to 50), while the y-axis is the response time in milliseconds. Our empirical results show a response time that is acceptable for most of the experiments, that is, with a value under 1000 ms. It is worth remembering that the value of this experiment will mainly depend on the network infrastructure and load between our testbed and the Twitter servers, which are bounded to the Internet services.

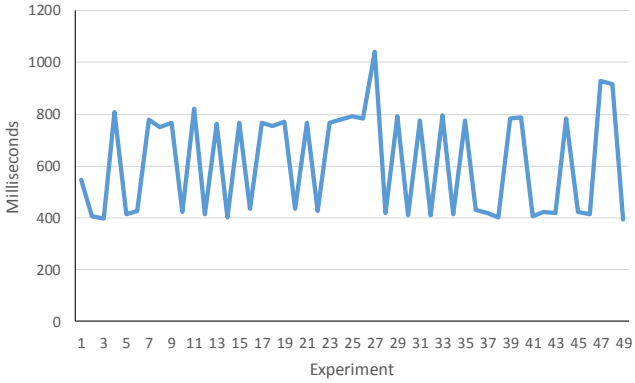

Figure 9: Response Time when Retrieving all the Direct Messages from a Specific Account in Twitter

VIII. Conclusions and Future Work

The usage of new technologies has increased exponentially with the rise of the Internet services. In the field of telephony, the classical telephony system, with its own network, is gradually replaced by VoIP, since it can use the existing data network (resulting in cost cut), and it allows a high degree of flexibility when developing custom applications.

Although unified communications systems and social networks have been around for some time, their possible integration has not been fully explored or exploited. In this research work, we proposed a solution that combines these two technologies together, to expand the communication channels offered to clients, for the customer service of an organization, by adding social networks as a new possible channel, to the range of tools currently offered by state-of-the-art telephone systems.

We are very interested in pursuing our work in this field. Since just a few researchers have been working in this direction, there are many open possibilities to enhance the area, and we are planning to explore the following ones:

- Transform the processing module into a backoffice, to have a simple and configurable interface.
- Develop a social network module where configurations can be made directly within Elastix, allowing the administration of outbound campaigns through direct messages or tweets to several customers using configurable templates.
- Produce predesigned templates to be used by the agent console when sending direct messages.
- Develop a statistics generator with a simple interface in terms of attended requests and their response times.
- Add artificial intelligence in the processing of messages, to train the system to identify negative words or phrases.
- Integrate some processes for data mining to know more about the organization's customers, with the goal of customizing outgoing campaigns.
- Port our development to Issabel [36][37], a fork of Elastix, maintained by the Internet community.

References

[1] G. Barajas Puente, *Elastix Unified Communications Server Cookbook*, Packt Publishing, March 2015.

[2] D. Duffett, *Getting Started with Elastix: A Beginner's Guide*, P8Tech, November 2013.

[3] N. Anaya, *Fundamentos de Telefonía IP e Introducción a Asterisk/ Elastix*, January 2013.

[4] P. Estrella, J. Bustos, and A. Muñoz, *Implementando Call Center con Elastix*, 2013.

[5] *VoIP Wiki*. http://www.voip-info.org.

[6] A. Peicevic, *Introduction to Asterisk: Learn How to Set Up your own PBX Telephone System*, 1st edition, CreateSpace Independent Publishing Platform, January 2017.

[7] T. R. Lewis, *High Availability Asterisk PBX: Proven Method*, 1st edition, CreateSpace Independent Publishing Platform, January 2016.

[8] D. Merel, B. Dempster, and D. Gomilion, *Asterisk 1.6 - Build Feature-rich Telephony Systems with Asterisk*, Packt Publishing, 2009.

[9] V. Stanislovaitis, *How to Start a VoIP Business: A Six-Stage Guide to Becoming a VoIP Service Provider*, 1st edition, Vilius Stanislovaitis, February 2016.

[10] J. Joch, *Openfire: Create your own XMPP Messaging Server Open Source Software*, CTS GMBH, January 2017.

[11] K. Thomas, *Email Architecture, Design, and Implementations*, 2nd edition, CreateSpace Independent Publishing Platform, April 2017.

[12] R. Hildebrandt and P. Koetter, *The Book of Postfix: State-of-the-Art Message Transport*, 1st edition, No Starch Press, March 2005.

[13] H. Alassouli, *Creation of Postfix Mail Server Based on Virtual Users and Domains*, Independently published, June 2018.

[14] N. Simionovich, *Asterisk Gateway Interface 1.4 and 1.6 Programming*, Packt Publishing, February 2009.

[15] B. Jackson, C. Clark, J. Long, and L. Chaffin, *Asterisk Hacking*, Syngress, June 2007.

[16] iCode Academy, *JSON for Beginners: Your Guide to Easily Learn JSON in 7 Days*, Independently published, August 2017.

[17] T. Marrs, *JSON at Work: Practical Data Integration for the Web*, 1st edition, O'Reilly Media, July 2017.

[18] A. Lombardi, *WebSocket: Lightweight Client-Server Communications*, 1st edition, O'Reilly Media, September 2015.

[19] D. Coward, *Java WebSocket Programming*, 1st edition, McGraw-Hill Education, September 2013.

[20] J. Del Moral, *Redes Sociales ¿Moda o Nuevo Paradigma?*, Madrid: Asociación de Usuarios de Internet, 2005.

[21] P. Siriwardena, *Advanced API Security: Securing APIs with OAuth 2.0, OpenID Connect, JWS, and JWE*, 1st edition, Apress, August 2014.

[22] A. Eloy Nascimento, *OAuth 2.0 Cookbook: Protect your Web Applications using Spring Security*, Packt Publishing, October 2017.

[23] J. Richer and A. Sanso, *OAuth 2 in Action*, 1st edition, Manning Publications, March 2017.

[24] S. Ludin and J. Garza, *Learning HTTP/2: A Practical Guide for Beginners*, 1st edition, O'Reilly Media, June 2017.

[25] M. Miller, *Sams Teach Yourself TweetDeck in 10 Minutes*, 1st edition, Sams Publishing, December 2010.

[26] H. Gaibor, *Integración de Elastix con Gtalk*. http://www.elastix.com/integration-of-elastix-with-gtalk.

[27] S. Smith, *Publicación en Twitter Mediante ASR*. http://www.10000horas.com/2011/12/08/un-agi-legendario-publicacion-en-twitter-mediante-asr.

[28] J. Coalson. *Free Lossless Audio Codec*. https://xiph.org/flac.

[29] L. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition*, 1st edition, Prentice Hall, April 1993.

[30] SoX, *Sound eXchange*. http://sox.sourceforge.net.

[31] *Ruby on Rails*. http://rubyonrails.org.

[32] *Silex: The PHP Micro-framework based on the Symfony Components*, https://silex.symfony.com.

[33] *RabbitMQ*. https://www.rabbitmq.com.

[34] G. Roy, *RabbitMQ in Depth*, 1st edition, Manning Publications, September 2017.

[35] E. Ayanoglu, Y. Aytas, and D. Nahum, *Mastering RabbitMQ*, Packt Publishing, January 2016.

[36] *Issabel*, https://www.issabel.org.

[37] C. Cabrera, *Issabel: Una Alternativa de Código Abierto*, https://asteriskmx.org/issabel-una-alternativa-de-codigo-abierto.