Improvement of the response time in an Open Source audioconference architecture based on SIP Multicast implemented with JainSIP, JainSDP and JGAP libraries

Carlos M MORENO
Communicacions Department, Universidad Central de Venezuela
Caracas, Distrito Capital PO 1051, Venezuela

and

Maribel ALVAREZ
Technological Studies Dean, Universidad Simón Bolívar,
Sartenejas, Estado Miranda PO 1086 , Venezuela

ABSTRACT

Group services like the audioconference require a minimum level of quality of service for multicast sessions. This work proposes a new overlay multicast architecture based on SIP extensions and a genetic algorithm. The architecture consists of a SIP Extender client (SE), a Multicast Gateway Agent (MGA) and a Multicast Manager (MM). The SE receives information about the most adequate MGA for it determined by a genetic algorithm inside the MM, then connects the chosen MGA and maintains connection with the MM itself. The genetic algorithm is implemented with JGAP(Java Genetic Algorithm Package) libraries. The SE and MGA are programmed with JainSIP and JainSDP libraries which contain Java structures associated with the SIP protocol and session description. Some experiments over UTP wired and WiFi IEEE802.11n network were performed. Partial results with static and dynamic MGA selection show that, if we compare the joining and leaving time measured inside a station containing SE client programmed with JainSIP and JainSDP libraries versus SJphone proprietary client, the software engineering may have more influence than the medium access method in the response time for a potential group member. Even more, the genetic algorithm at the MM minimizes the response time at great scale.

Keywords: Open source, Audioconference, SIP Multicast, JainSIP

1. INTRODUCTION

Nowadays voice, video and data traffic are getting integrated in the same networking platform. The design of such architecture has to take into account the applications to be deployed considering the highest layers of the TCP/IP model [1]. One of the most demanded applications is the audioconference between groups, and its improvements is being done over applicative layer. An adequate level of quality of service is required for these applications, like a low response time for joining and leaving group member, and this can be achieved through the use of multicast [2][3]. There are three types of multicast: IP multicast, Overlay Multicast (OM) and Application Layer Multicast (ALM). IP Multicast is used in IPTV with the use of protocol IGMP (Internet Group Management Protocol) for IPv4, or MLD (Multicast Listener Discovery) for IPv6 [4]. The second approach uses the end systems and intermediate proxies to form the trees [5]. In the third approach the end systems are completely responsible for the creation and destruction of trees [6].

A new architecture for audioconferences based on overlay Multicast with several clients has been designed and implemented [7]. Such architecture consists of an extended SIP [8] client, a multicast manager and a MGA (Multicast Gateway Agent). The multicast manager module has been enhanced with APIs JainSIP [9] and JainSDP [10] open source libraries. At great scale it is possible that the response time depends not only on the SIP timers and delays [11] of the network but also on the process to find the best multicast trees inside the multicast manager. That is the reason why we propose the use of a genetic algorithm. Three test groups were mounted to compare the response time over UTP wired versus over WiFi, and to analyze an isolated JGAP based genetic algorithm at MM.

In section 2, we explain the relationship between SIP protocol and multicast. Then, in section 3 we describe JainSIP and JainSDP libraries for SIP programming. After, in section 4 we explain the relationship between genetic algorithms, open source and multicast, and describe JGAP libraries. In section 5 we explain details of the new architecture. In section 6 we explain the experimentation done in the three tesbeds to compare them. In section 7 we show the results of measured response time on the new architecture, and finally in section 8 we explain the conclusions and perspectives.

2. SIP AND MULTICAST

.

IP multicast can be used with SIP as a discovery-like service for a simple host, where it sends a simple request to a group of homogeneous servers, and processes the response of only one of them. This functionality is mainly used for registrations with multiple servers[12].

Some work has been done since 1998 to allow multicast and unicast conference sessions with SIP. In that year, an article published by both Dr. Schulzrinne and Dr. Rosenberg show how IP multicast conferences could be established[13]. This approach presents the drawbacks of IP multicast in a LAN or WAN. It forces the network administrator to install mrouters, and switches that support dynamic group management protocols like IGMP snooping. As a consequence, two more approaches are used: Overlay Multicast and Application Layer Multicast. Overlay Multicast permits that proxies are located in the center of the trees topology. In the ALM approach, the end systems are responsible for the creation, maintenance and termination of

multicast groups. A new architecture named SIP Multicast for audioconferences based on Overlay Multicast was designed at small scale in 2009 [7]. The present work improves the initial prototype by introducing additional software engineering APIs, and optimization, considering also its performance over WiFi.

3. SIP AND OPEN SOURCE

Several projects have been created by the community supporting SIP protocol. For instance, ASTERISK[14] and its precompiled version ELASTIX, consists of an IPBX server that provides basic signaling functions and more complex features like centralized conference, instant messaging, presence, redirectioning among them. Some projects located at the end systems as softphones like JITSI were programmed with Java language. On this specific case, JITSI is based on open source libraries JainSIP (JSR032) and JainSDP (JSR031) created by NIST (National Institute of Standards and Technology). Other libraries are SIP servlets (JSR289), SIMPLE instant messaging (JSR165) and Jain SLEE (JSR22 and JSR240). This work is based on JainSIP and JainSDP libraries.

JainSIP
JainSIP (Java APIs for Integrated Networks SIP) is a Java standard for a low-level SIP interface. It provides access to SIP at the lowest level. Its programming constructs represent concepts such as messages, headers, parameters, ports, and IP addresses.

JainSDP
Defines a Java Interface to facilitate manipulation of SDP (Session Description Protocol) which describes SIP sessions within requests and response messages.

4. GENETIC ALGORITHMS AND OPEN SOURCE

Several open source projects have been created to implement metaheuristic methods. In the area of genetic algorithms one of the best documented projects is JGAP (Java Genetic Algorithm Package).

JGAP
Based on Java language, JGAP contains classes and methods that represent the natural evolution[15]. The programmer defines basic parameters like: number of population members, number of generations and genetic operators (mutation, crossover). It is possible to add new genetic operators and define a customized fitness function. Some studies suggest the use of concurrent programming for a faster execution.

GA AND MULTICAST
Multicast technologies are based on the creation of trees represented by graphs. Optimization techniques can be employed to generate and choose the most adapted tree according to the problem. Genetic algorithms are under research just to find the best graph. Its velocity depends on the software engineering and design of the algorithm.

GA and IP Multicast: Bhattarchya & Venkaterwaram (2005)[16] proposed a new IP multicast routing scheme based on GA. Results showed that simulations of such metrics, topologies and number of nodes ranging from 32 to 1000, yield better performance than existing multicast routing schemes.

GA and Overlay Multicast: Wan & Gang (2008) [17] proposed a heuristic genetic algorithm for multicast overlay network link selection that reduces the overlay network links by using fast converging speed and stabilization with simply operation.

GA and Application Layer Multicast: Pen & QionGhau (2005) [18] proposed a novel application that includes load balance constraints at both application and network layer within an adapted fitness function.

5. SIP MULTICAST ARCHITECTURE

The new architecture SIP Multicast based on SIP protocol, with Overlay Multicast and groups management consists of a SIP extender (SE), a Multicast Gateway Agent (MGA) and a Multicast Manager(MM).

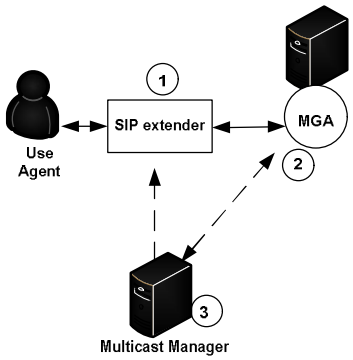

Fig. 1 SIP multicast architecture

SIP Extender-SE: this component permits to work with SIP extended signaling. It adds an additional header field that indicates the use of Overlay Multicast. It also converts back extended messages with SIP multicast into SIP conventional ones.

Multicast gateway Agent-MGA: it interprets the extended SIP multicast messages and responds to the SE user agents. It fills out and reads information from an Applicative Multicast Table that contains information about the user agents associated with their respectives SIP extenders.

Multicast Manager-MM: consists of the brain of the SIP multicast architecture and works only on the control plane. A human operator can choose whether a specific user agent will solicite its inclusion or exclusion for an audioconference if a static configuration is done. In the automatic modality, it can calculate the optimal topology. It means to find out which MGA must be connected with the SIP extender.

SE and MGA area programmed based on JainSIP and JainSDP APIs. MM is programmed with JGAP APIs and multithreading.

6. EXPERIMENTATION

We implemented three testbeds just to measure joining time , and leaving time on the system fixed vs wireless, and processing time within MM at great scale.

Joining Time

We used two criteria as Joining time:

$$JT1 = T_2 - T_1 \qquad \text{Eq.(1)}$$

Where:

JT1: Joining Time

T_2: Instant when the first media packet arrives

T_1: Instant when the INVITE extended message is sent

$$JT2 = T_3 - T_4 \qquad \text{Eq.(2)}$$

Where:

JT2: Joining Time

T_3: Instant when ACK extended message is sent

T_4: Instant when INVITE extended message is sent

Leaving Time

We used two criteria as Leaving time:

$$LT1 = T_6 - T_5 \qquad \text{Eq.(3)}$$

Where:

LT1: Leaving Time

T_5: Instant when last media packet arrives

T_6: Instant when BYE extended message is sent

$$LT2 = T_7 - T_8 \qquad \text{Eq.(4)}$$

Where:

LT2: Leaving Time

T_8: Instant when BYE extended message is sent

T_7: Instant when OK extended message is received

Processing Time

Processing Time is considered as follows:

$$PT = ET - ST \qquad \text{Eq.(5)}$$

Where:

PT: Processing Time measured with TimeMesurer software

ST: Starting Time before applying GA

ET: Ending Time after applying GA

Testbed 1 Joining and leaving Time fixed network:

Consists of two softphones including SIP Extenders (SE) each, a PC with MGA and MM modules inside, and a router WiFi WRT160N. MGA is implemented with JainSIP and JainSDP. MM is implemented with Java. A genetic algorithm based on JGAP was isolated. All terminals are connected via UTP cable (Fig. 2).

TimeMeasurer software was used to calculate PT. The softphones studied are JITSI and SJphone. Wireshark™ was used just to get JT1, JT2, LT1 and LT2.

Joining Time and Leaving Time are measured 10 times by using Wireshark™ software. This testbed was located in a closed room.

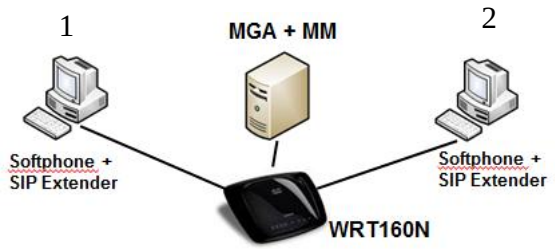

Fig. 2 Fixed network testbed

Testbed 2 Joining and leaving Time WiFi: consists of two softphones + 2 SIP Extenders (SE), a PC with MGA and MM modules inside, and a router WiFi WRT160N. MGA is implemented with JainSIP and JainSDP. MM is implemented with Java. A genetic algorithm based on JGAP was isolated. MGA and MM are connected to the router via UTP cable, and the terminals with the softphones via WiFi IEEE802.11n (Fig. 3). The softphones studied are JITSI and SJphone.

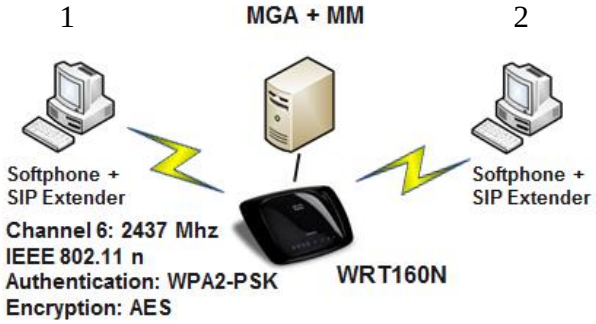

Fig. 3 Wireless network testbed

Joining Time and Leaving Time are measured 10 times by using Wireshark™ software. This testbed was located in a closed room.

Testbed 3 Processing time in isolated MM: c

Consists of the GA searching operation within MM based on JGAP with multithreading [19]. The population contains 80 individuals each one formed by a chromosome of one gene with maximum 32 bits (Integer). The initial population was randomly created and 500 generations were employed on the evolution. The best individual was calculated 10 times with 5,12,19,26 and 32 bits gene representing a graph that contains a potential MGA.

We emulated the process of selecting the best MGA by using the fitness:

$$F = \frac{100}{D_{SM} + D_{MGM} + D_{MMS}} \qquad \text{Eq. (6)}$$

Where:

F: Fitness Function of GA.

D_{SM}: Weight representing lowest bandwidth of link between SE and potential MGA.

D_{MGM}: Weight representing lowest bandwidth of link between potential MGA and MM.

D_{MMS}: Weight representing lowest bandwidth of link between MM and SE.

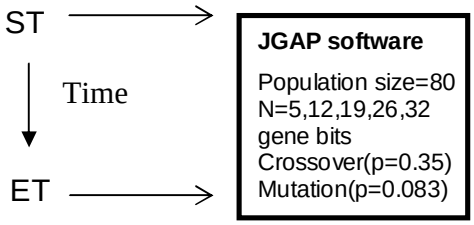

Fig. 4 Processing Time testbed

7. RESULTS

Joining Time JT1 and Leaving Time LT1

Considering the measured variable JT1 (Fig. 5) , it can be seen that its highest value corresponds when JITSI software was employed mainly, with UTP connection or with WiFi access medium. The central PC contained modules MGA and MM. MGA location is predefined on this case without a genetic algorithm in a static configuration.

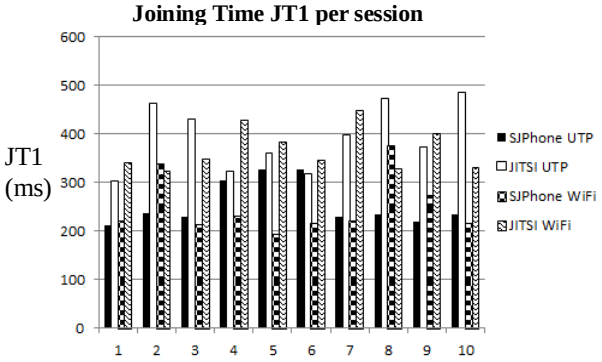

Fig. 5 Joining Time JT1

Considering the measured variable LT1 (Fig. 6), it can be seen that its highest value corresponds when SJphone software was employed mainly, with UTP connection or with WiFi access medium. The central PC contained modules MGA and MM. MGA location is predefined on this case without a genetic algorithm in a static configuration.

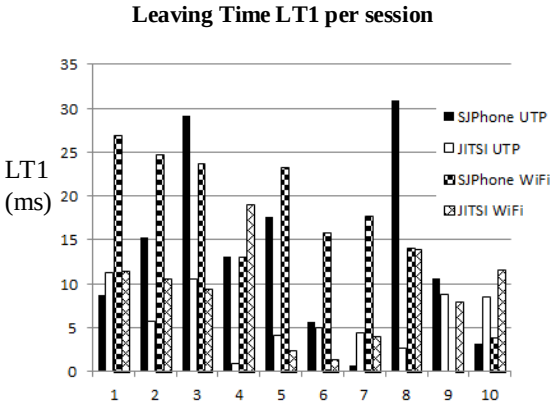

Fig. 6 Leaving Time LT1

Joining Time JT2 and Leaving Time LT2

Considering the measured variable JT2 (Fig. 7) , it can be seen that its highest value corresponds when SJphone software was employed mainly, with UTP connection or with WiFi access medium. The central PC contained modules MGA and MM. MGA location is predefined on this case without a genetic algorithm in a static configuration.

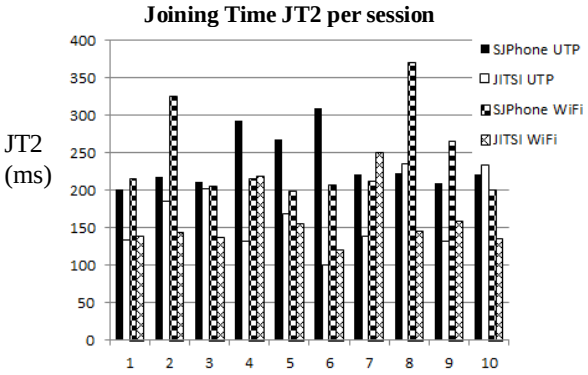

Fig. 7 Joining Time JT2

Considering the measured variable JL2 (Fig. 8) , it can be seen that its highest value corresponds when SJphone software was employed mainly, with UTP connection or with WiFi access medium. Two peaks were presented with WiFi technology. The central PC contained modules MGA and MM. MGA location is predefined on this case without a genetic algorithm in a static configuration.

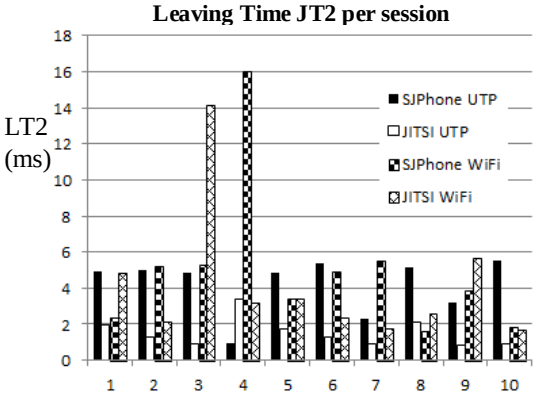

Fig. 8 Leaving Time LT2

The Fig. 9 compares the average JT1 and JT2 values.

Joining Time JT1 and JT2 comparison

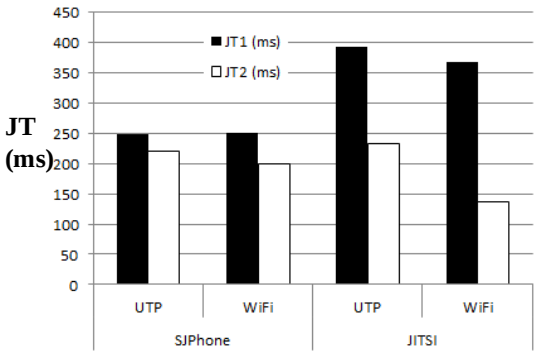

Fig. 9 JT1 vs JT2

In Fig. 9 we can see that average value of JT1 is higher with JITSI which uses JainSIP and JainSDP. However, it can be seen that JT2 has its lowest value with JITSI and WiFi. We can note that, software engineering has more impact than access medium for JT1. It is possible that access medium has more impact than software engineering for JT2; but it can not be assured for the scenarios studied because the average differences are not so high.

Leaving Time LT1 and LT2 comparison

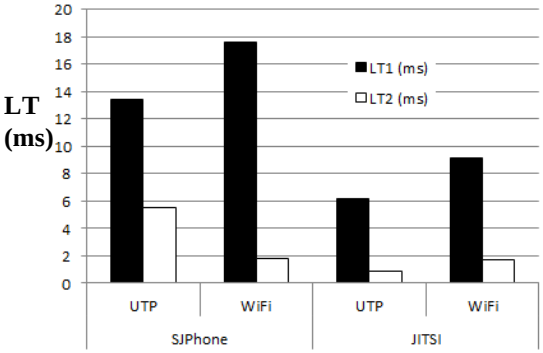

Fig. 10 LT1 vs LT2

In Fig. 10 we can see that average value of LT1 is higher with SJphone. However, it can be seen that LT2 has its lowest value with JITSI and UTP. It is possible that access medium has more impact than software engineering for LT2; but it can not be assured for the scenarios studied because the average difference is not high.

Processing Time with GA

Considering the measured variable PT (Fig. 11) in an AG within isolated MM module, we can notice that PT is decreased from 2^5(32) through 2^{26} (67108864) potential MGAs. For 2^{32}(4294967296) potential MGAs the PT is also high.

Processing Time PT versus N⁰ of gene bits

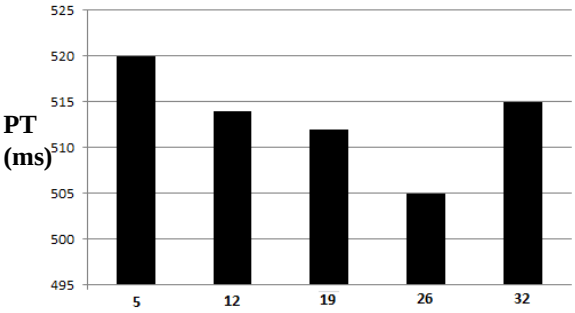

Fig. 11 Processing Time vs N⁰ of bits per gene

As the number of potential MGAs is increased until 2^{26}(67108864) for a multipoint audioconference, a lower PT would produce fewer retransmissions of SIP extended messages, so improving JT1, JT2, LT1 and LT2 at great scale with UDP protocol. In the case of 32 bits gene, PT is higher than 500 ms and so SIP extended messages delays no matter if access method is fixed or wireless. According to RFC3261[12] 500 ms is the default value for timeout in SIP responses over UDP. As a consequence, with a 32 bits gene there will be several SIP retransmissions.

8. CONCLUSIONS AND PERSPECTIVES

On the scenarios studied for SIP multicast architecture, software engineering has more impact than access medium if we compare the open source JITSI made with JainSIP and JainSDP versus the proprietary software SJphone. If we include a genetic algorithm made with JGAP inside MM, there would be SIP extended messages retransmissions as processing time would be higher than 500 ms which is comparable to the default value of maximum delay in provisional responses if UDP protocol is employed.

However, if we increase the number of potential MGAs from 2^5 (32) through 2^{26}(67108864) the processing time within MM when using a genetic algorithm is decreased, and so the probability of existing SIP extended messages retransmissions.

If we choose criteria JT1 and LT1 based of media and control plane to study response time, it permits discriminate more easily the impact of software engineering versus access medium than if we use criteria JT2 and LT2 based only on control plane.

The future work focuses on inclusion of the genetic algorithm made with other programming languages within MM, the study of the SIP Multicast architecture with several SEs by using a simulation tool, the behavior of the architecture in high congestion scenarios at great scale with WiFi distanced elements, and study of retransmissions with SIP extended messages over both UDP and TCP transport protocols.

9. ACKNOWLEDGEMENTS

Tha authors thank to the Consejo de Desarrollo Científico y Humanístico (CDCH) of the Universidad Central de Venezuela for the Budget assigned to the project: “Mejoramiento de arquitectura para servicios multimedia basada en multidifusión IP” under the identification PI-08-7820-2009/1.

10. REFERENCES

[1] R. Winter, **"Towards a Unified Internet Control Architecture"**, Proceedings of 8th Workshop on IP (Euroview2008), Würzburg, Germany, jul-2008.

[2] Beau Williamson, **Developing IP Multicast Networks. Volume I**, 1st ed. Indianapolis: CISCOPRESS, 2001.

[3] R. Seifert, **The Complete Guide to LAN Switching Technology**, 1st ed. USA: Willey, 2001.

[4] Daniel Minoli, **IP Multicast with Applications to IPTV** *and Mobile DVB-H*, 1st ed. USA: Willey-Interscience, 2008.

[5] J. Jannotti., D. Gofford., K., Johnson.,M. Kaashoek., J. O´Toole, **"Overcast: Reliable Multicasting with an Overlay Network"**, *Proc. 4th Symp. Operating Systems Design and Implementation (OSDI).*

[6] D. Pendarakis., S. Shi ., D. Verma., M. Waldvogel, **"ALMI: An Application Level Multicast Infrastructure"**, *Washington University*, sep-2000.

[7] C. Moreno., M. Becker., P. Vincent, **"Performance of a new multicast architecture based on SIP extensions and group management"**, IEEE Latin American Network Operations and Management Symposium, 2009. LANOMS 2009., Punta del Este, 2009, pp. 1-10.

[8] Gonzalo Camarillo, **SIP demystified,** 1st ed. USA: McGraw Hill, 2002.

[9] **"An Introduction to the JAIN SIP API"**. [On line]. Available at: http://www.oracle.com/technetwork/articles/entarch/introduction-jain-sip-090386.html. [Consulted: 01-jun-2014].

[10] **"The Java Community Process(SM) Program - JSRs: Java Specification Requests - detail JSR# 141"**. [On line]. Available at: https://jcp.org/en/jsr/detail?id=141. [Consulted: jun-01-2014].

[11] Perea, Rogelio, ***"Internet Multimedia Communications Using SIP"***., 1.ª ed. USA: Morgannn Kauffma, 2008.

[12] J. R. H.Schulzrinne., G.Camarillo ., A., Johnston ., J. Peterson., R.Sparks., M. Handley., E.Schooler, **"SIP: Session Initiation Protocol"**, 2002.

[13] H. Schulzrinne ., J. Rosenberg, **"Signaling for Internet Telephony"**, *Proc. of IEEE Sixth International Conference on Network Protocols*, pp. 298-307, oct-1998.

[14] **"Linux Asterisk Zone "** Blog Archive » Protocolos VoIP - Códecs», 25-sep-2008. [On line]. Available at: http://www.linuxasteriskzone.com. [Consulted: sep-25-2008].

[15] **"JGAP: Java Genetic Algorithms Package"**, 10-feb-2014. [On line]. Available at: http://jgap.sourceforge.net/. [Consulted: may-31-2014].

[16] R, Bhattacharya ; P,Venkateswaran ; S, Sanyal, **"Genetic Algorithm based efficient routing scheme for multicast networks"**, IEEE International Conference on Personal Wireless Communications ICPWC 2005, Colmar, Francia, 2005, pp. 500-504.

[17] D. Wang ; J. Gan ; D. Wang, **"Heuristic Genetic Algorithm for Multicast Overlay Network Link Selection"**, presentado en Second International Conference on Genetic and Evolutionary Computing WGEC 2008, Jingzhou, Hubei , China, 2008, pp. 38-41.

[18] C. Peng ; D. QionGhai ; W.Qiufeng, **"An application layer multicast routing algorithm based on genetic algorithms"** Proceedings of the 8th International Conference on Telecommunications. ConTEL 2005, Zagreb, Croatia, 2005, vol. 2.

[19] C. Moreno., M. Alvarez, **"Optimization of the response time in a multicast audioconference by using a genetic algorithm and concurrent programming"**, XII Congreso Internacional de Métodos Numéricos en Ingeniería y Ciencias Aplicadas, CIMENICS2014 Pampatar, Margarita, Venezuela 2014, p. OP 37-42.