Performance Analysis of a Raspberry Pi Based IP Telephony Platform

Peláez D.*; Estrada J. A.; Tipantuña C.**; Estrada J. C***;**

**Escuela Politécnica Nacional, Instituto Geofísico, Quito, Ecuador*
e-mail: dpelaez@igepn.edu.ec
*** Escuela Politécnica Nacional, Facultad de Ingeniería Eléctrica y Electrónica, Quito, Ecuador*
e-mail: {jose.estrada; christian.tipantuna}@epn.edu.ec
****email: jancoej@gmail.com*

Resumen: La telefonía es un servicio que se usa muy ampliamente todavía, ya que con éste han crecido las últimas generaciones de personas. La integración de la telefonía con Internet y la posibilidad de implementarla a nivel de *software* le ha dado a este servicio una flexibilidad asombrosa para el despliegue de sistemas de comunicación de voz. Además, gracias al desarrollo de código abierto, un sistema PBX puede instalarse ahora en casi cualquier dispositivo de cómputo.Este es el caso de la placa Raspberry Pi, una plataforma de cómputo pequeña y de muy bajo costo que actualmente está siendo usada para distintas aplicaciones en entornos domésticos, y que es compatible con sistemas operativos basados en Linux. Así, el objetivo de este artículo es evaluar el rendimiento de un sistema de Telefonía IP basado en la plataforma Raspberry Pi de acuerdo a parámetros objetivos como uso de CPU, uso de memoria RAM, tiempos de respuesta de llamada y llamadas fallidas. Sin embargo, se toman en cuenta también parámetros objetivos de evaluación, mediante la realización del *test* MOS, con el fin de medir la percepción del usuario sobre el servicio telefónico en un entorno específico utilizando la plataforma mencionada. Encontramos que la plataforma Raspberry Pi puede soportar varias llamadas concurrentes sin que se sobrecarguen sus recursos pero que, en determinado punto, los tiempos de respuesta se incrementan significativamente, así como el número de llamadas fallidas.Considerando el costo reducido de la placa debido a la no utilización de licenciamiento a nivel se software o hardware y tomando en cuenta el rendimiento obtenido con diferentes valores de carga, se evidencia como una interesante opción para pequeñas y medianas empresas, así como para entornos domésticos de aplicación.

Palabras clave: Telefonía IP, Raspberry Pi, test MOS, prueba de carga, rendimiento

Abstract: Telephony is still heavily used since it is a service with which people are very familiar. Its integration to Internet and its implementation at the software level have given this service an amazing flexibility when deploying voice communication systems. Moreover, thanks to the open source development, a PBX service can now be installed in almost every computing device. That is the case of Raspberry Pi low-sized and very cheap computing platform that is being employed to provide network services for SOHO environments, and which is compatible with Linux based operating systems. The aim of this paper is to assess the performance of a Raspberry Pi based IP telephony service in terms of objective parameters such as CPU and RAM usage, call response times, and failed calls. However, subjective parameters, in terms of user perception about the service, were also taken into account by applying the test MOS in a specific test environment. We found that the codec selection and the simultaneous number of calls determine the overloading point of the platform (in terms of CPU usage) which was evidenced by longer response times and failed calls. Considering the reduced cost of the board, since no hardware or software licenses are required, and taking into account the performance obtained from the system with different load levels, this is a very interesting option for small and medium companies, and also for domestic application environments.

Keywords: IP telephony, Raspberry Pi, MOS test, load test, performance

1. INTRODUCTION

As a result of the exponential growth of Internet penetration (and subsequent price reduction), both organizations and end users are increasingly using IP based technology to satisfy their communication requirements. One of such requirements is voice communication through telephony service, which was one of the first and most popular services provided over information networks [4]. Thus, Voice over IP solutions are not only being implemented in big companies but also in small businesses. Indeed, IP telephony systems can now be installed in domestic environments due to the lower costs of accessing to technology [2].

Raspberry Pi is a tiny, low-sized, and very cheap computing platform that is being used to deploy applications of different kinds [1]. Its small size allows greatly reduced energy consumption which lowers even more the overall cost of Raspberry Pi based solutions. This device, which is based on ARM architecture, works with Linux-based operating systems, so many open source software products can be compiled to run network services, such as IP telephony.

Although the Raspberry Pi platform is being tested with many interesting projects, it is not clear yet how some intensive applications would impact on the overall performance of this hardware device. IP telephony services involve some CPU demanding processes, such as digitalization, codification or transcoding of voice.

The contribution of this work consists of showing how usefulsmall-sized hardware combined with open source software is to provide small business voice services. Given the current technological dependence of some countries, efficiently using cheap solutions would help such countries reduce this technological gap. Consequently, this study pretends to evaluate the performance of a Raspberry Pi platform when providing very simple IP telephony services (phone calls) over an information network. The methodology we used includes creating a plain network to connect the Raspberry Pi platform with a voice terminal instance. Load testing is done from the terminal instance to the IP phone solution in order to determine the limits at which the service gets overloaded. Sipp was used to dynamically generate calls through an Asterisk based IP telephony service which was installed and configured in the Raspberry Pi platform.

The rest of the paper is organized as follows: in section 2, we describe some background concepts about the Raspberry Pi computing platform, the µElastix open source solution for Unified Communications, and some voice codecs used to digitalize the sound transmitted over a network. In section 3, we present the scheme and the methodology used to do the performance tests over the prototype we built. The results of the performance analysis are depicted in section 4, and in section 5 conclusions are drawn.

2. BACKGROUND

2.1IP Telephony service

Telephony is an application technology whose major aim is the transmission of voice between two terminals. When the voice is digitized and compressed (adapted) to be sent over IP networks, we refer to IP telephony. Voice over IP (VoIP) and telephony over IP (ToIP) are terms commonly used interchangeably. ToIP mostly refers to the telephony service, all its infrastructure (IP phones, gateways, IP PBX systems) and the signaling and real-time protocols necessary to initiate, release a phone connection or to provide additional services over a telephone network. Nevertheless, VoIP only denotes the mere transmission of voice over IP networks.

The core of an IP telephony system is the IP PBX, which is a central device that concentrates (and distributes) all the voice communications among the user terminals (phones, softphones, etc.). In order for these terminals to initiate or end a call, signaling protocols are implemented in both the terminals and the IP PBX. SIP and H.323 are the most popular standard signaling protocols.

When a call is established, the media (sound, mainly) is transported using session-layer real-time protocols (RTP and RTCP). These protocols are responsible for providing the communication process with the mechanisms to effectively transmit the media over the network. Codecs convert the analog voice (captured from a person) to digital bit streams, and may compress these flows at different rates. The quality of a call and the hardware resources consumed (network, processing, RAM) basically depend on the codec used.

A wide range of codecs are available, which may offer more or less quality, in terms of sound, compression, or bandwidth consumption, and they may (or not) require a license. There is a pretty known tradeoff among these parameters that must be taken into account when choosing the codec. In general, high compression ratios (low bandwidth usage) derives in intensive usage of CPU. Even more processing resources are needed if a very compressed flow has to be provided with high sound quality. The codec offering the best benefit-cost ratio is G.729 which is licensed. G.711 hardly compresses the voice (in the *companding* process) so avoids losing information in the processes of digitalization. GSM, G.722, and iLBC are unlicensed codecs. An analysis of the behavior of these codecs is done in [11].

2.2 Asterisk: The Open Source IP Telephony Implementation

Asterisk [9] is an open source (Linux based) telephony switching service, which implements plenty of the facilities of an IP PBX system. Since it is very popular among the enthusiasts of the IP telephony, some Linux distributions based on Asterisk have appeared that are oriented to provide a complete set of IP telephony services. Elastix [6] is one of such distributions that incorporates a web based graphical interface for the user to be able to easily configure the embedded Asterisk daemon. A version of Elastix called µElastix [5] is compiled to work on ARM devices such as the Raspberry Pi.

2.3 Raspberry Pi Platform

Raspberry Pi [1] is a tiny, low-sized, and very cheap computing platform that was created to help students to learn computer science. It includes a 700 MHz (or more) processor, GPU, RAM (up to 512 MB), and SD or micro SD sockets for persisting storage. A Raspberry Pi platform also offers USB ports, audio/video input/output, an Ethernet interface, and low-level peripheral capabilities.

Currently, the power of its processor is compared to the power of old processors that were sold few years ago (Pentium 3 or 4) and, given its very small size and therefore low energy consumption, it offers a great benefit-cost ratio. This is the reason for which it is becoming an interesting alternative to

deploy small business or home network services at a very low cost.

The Raspberry Pi platform supports a great deal of lightweight Linux distributions such as: Raspbian, Pidora, Arch Linux ARM, etc., which means that many Linux based services can also be installed. Several specific purpose distributions are also been ported to ARM so that they can run on Raspberry Pi. Solutions for IP telephony, firewall, media center and centralized storage have been compiled to work on Raspberry Pi.

2.4Performance Metrics

Many parameters can be measured in order to evaluate the performance of a phone system, which is, in fact, a computer system. This computer system is certainly connected to a network, together with the terminal devices and, probably, to the Internet with the purpose of providing voice communication services, not only within the local network. Consequently, the bottlenecks of the phone service are the IP PBX and the capacity of the Internet connection.

Among the tasks performed in the IP PBX, encoding and decoding are the most CPU intensive processes that maintain the capacity of an IP telephony platform bounded [10]. The codecs are responsible for encoding/decoding the voice and thus the codec used effectively determines how much CPU is going to be used. Since the encoding/decoding is done on a per-link basis, the processor has to do this work for each communicating link established among the IP PBX and each terminal. When transcoding is done (two terminals in a call using different codecs) the CPU load over the IP PBX system significantly increases. Clearly, the processor capacity has to be sized based on the codecs used with concurrent calls in the worst case (peak hour).

The bandwidth needed for each voice channel may also limit the capacity of the phone system [12], especially through a (commonly) restricted Internet connection. If higher fidelity is required, probably more bandwidth would be needed to transmit more information (not so compressed) through Internet. The capacity of the Internet connection has to be dimensioned according to the number of concurrently needed voice channels.

Finally, a very interesting approach to measure the performance of a voice communication service is by means of evaluating the perception the users have of such service. For this, the ITU-T (International Telecommunication Union - Telecommunications) has defined the Mean Opinion Score test (MOS) which is specified in the recommendation P.800 [8].

The documentation of this recommendation outlines a series of parameters under which the test has to be performed. The MOS is used to score the quality of a call based on subjective tests (the user's opinion). The ratings of the test are obtained from a set of questions answered by the users and such rating range from 1 (bad quality – very annoying experience) to 5 (excellent quality).

3. METHODOLOGY OF PERFORMANCE ANALYSIS

3.1 Performance Metrics Used

We chose to perform an automated stress test in order to evaluate the performance of a Raspberry Pi based telephony platform. Since this analysis is a preliminary approach, we only measured the CPU and RAM load generated by the work of different codecs (encoding/decoding) when simulated calls were established through the telephony.

Table 1. Technical specifications of Raspberry Model B

Systemon a Chip	Broadcom BCM2835
CPU	ARM (700 MHz)
GPU	Broadcom VideoCore IV
SD RAM	512 MB
USB Ports	2
Integrated Storage	SD / MMC
Network	10/100 Ethernet

However, provided that an overloaded system (a processor in this case) begins to lag in running processes, or even to discard them, two other metrics on which we focused were: response times and failed calls. Response times (times taken for the PBX to answer the call) increase as the load on the system increases and, ultimately, calls begin to fail (are never answered) when the system gets overloaded.

Provided that limited objective metrics can barely show details about the user experience with the voice communication system, we also performed a very simple MOS test to illustrate the user's perception about our prototype.

3.2 Evaluation Scheme and Tools

In order to test the capacity of the IP telephony system, we used the Model B Raspberry Pi board (technical specifications described in Table 1), and we connected it to a laptop from which we initiated the automated stress test. We used µElastix to implement the IP telephony system in the Raspberry Pi board, and Sipp [13] (installed in the laptop) as the client instance that generates a number of calls to stress the phone system.

Sipp is a load testing tool for IP phone systems. It only works with SIP protocol and it is capable of generating SIP messages to initiate and release a voice call through an IP PBX. Moreover, Sipp is able to send media (voice or video) when a call is established. It is very flexible when executing automated performance tests and provides extensive statistics during and after these tests. The scheme we used for each test consisted of generating maximum 200 calls from Sipp to the Raspberry Pi IP PBX system. The call generation process

starts with 2 simultaneous calls to reach a maximum number simultaneous calls, each of which lasts about 10 seconds. The test finishes when the 200 calls are generated. During each call we sent media (that was previously generated using a softphone and captured by means of Wireshark) using G.711, GSM, G.722 iLBC codecs. For each codec we made 9 tests, varying the number of simultaneous calls since 2 to 18 in steps of 2.

On the telephone system side, we configured a simple call flow that was executed each time a new call was received. Basically, the call is answered and then some music is played during 10 seconds. Finally, the call is hanged up. The whole scheme is illustrated in Fig. 2.

In order to measure the user perception about the Raspberry Pi based IP telephony platform, we built another testing scheme. It consisted of the already described Raspberry Pi platform providing IP telephony service through µElastix. This time, we loaded the system with 4 simultaneous calls generated by a Sipp instance which also sent voice compressed with GSM codec. At the same time, we wirelessly connected the Raspberry Pi together with two smartphones installed with the ZoIPer softphone. That way, for each test (3 minutes long), two users were able to connect by phone from separated rooms to then score their experience using the parameters defined in the ITU-T P.805 recommendation. A total of 14 tests were done, giving 28 filled surveys.

4. RESULTS AND DISCUSSION

In this section we discuss the results obtained from the performance tests done on the phone system (described in section 3.2). The information used to make the performance analysis was obtained from the traffic statistics provided by Sipp. Moreover, we also measured the processor and RAM usage by employing the tool htop [7].

4.1 CPU and RAM Usage

As expected, the hardware resources required for call processing are not dependent only on the codec used to compress the voice, but especially on the concurrent load that the system receives in terms of number of (simultaneous) calls. We measured the CPU and RAM usage of the Raspberry Pi platform when employing each of the codecs mentioned in the section 3.2 for the scheme proposed. As we explained before, we performed the load test using different amounts of concurrent calls.

Results (Fig. 1) show that the CPU usage increases uniformly with the amount of simultaneous calls, regardless of the codec used. Additionally, in terms of CPU usage, GSM is the less demanding codec, unlike iLBC and G.729 which are the ones that require more processing work than the others. It seems reasonable given the high quality offered by the latter codecs. Although theoretically G.711 is not a high CPU intensive codec (since it "only" does companding"), it does not show the best performance in terms of CPU usage. In general, the percentage of processor usage reaches values greater than 60% when more than 15 calls are handled concurrently by the Raspberry Pi platform.

With regard to the usage of RAM, the results show that the Raspberry Pi platform requires less than the 10% of the available memory (512 MB) to manage about 20 simultaneous calls through the IP telephony service (Fig. 3).

4.2 Response Times and Failed Calls

Even though the results obtained about CPU and RAM usage reveal interesting behaviors related to the codecs used, they may not represent enough evidence about the real performance of the telephone system.

Based on the statistics generated by Sipp, we were able to collect information on the response times taken by the IP phone system to answer each incoming call. As depicted in Fig. 5 (and it happens for every codec), for a given maximum number of concurrent calls, the response times raise up to a point where they get stabilized. However, when the system begins to overload, its response times vary nonuniformly (see Fig. 4) since some calls cannot be processed anymore. That is when the calls begin to fail and we could also collect such statistics that are depicted in Fig. 6. Consistent with the results described in the last subsection, the codecs requiring more processing power are the ones provoking more failed calls as illustrated in Fig. 7. It can be seen that calls start to be dropped because of 10 simultaneous calls through iLBC.

4.3 Mean Opinion Score (MOS)

In conjunction with the last scheme defined in the subsection 3.2, we tried to get a general idea of the users' opinion about the telephony system. Provided that it was the one with the best benefit-cost ratio, we tried out the GSM codec to perform a MOS test. The system, as described in the last section, got a score of 3.917 (good quality) from the 28 users that tested it. This result is consistent with the MOS values found in the literature [12], and it is excellent, considering the hostile environment of the wireless network we used.

5. CONCLUSIONS

This work is a first approach to measuring the performance of an IP telephony system deployed on a Raspberry Pi platform. Seemingly, the platform does a very good job processing calls since the percentage of processor usage did not exceeded 80 % even for 20 simultaneous calls. However, when evaluating specific application parameters, such as response times and failed calls, the actual state of the phone system begins to arise. Response times significantly increase (up to 500 ms) when more than 14 simultaneous calls are processed. G.729 works very well (even better than G.711 and GSM) in terms of response times, but GSM is the best when evaluating failed calls. iLBC and G.722 provoke call dropping when 10 and 14 simultaneous calls are generated, respectively. A more subjective MOS test showed satisfactory results for the GSM codec and 5 simultaneous calls despite the fact that the test was performed on a wireless network.

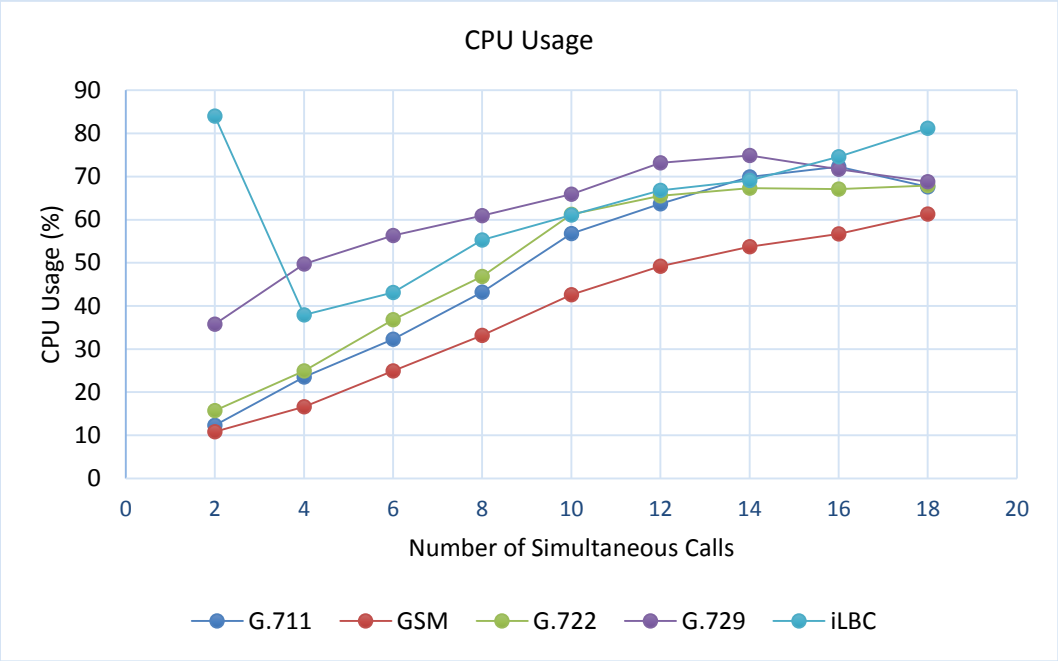

Figure 1. Percentage of CPU usage according to the codec used and the load of the system in terms of the number of simultaneous calls

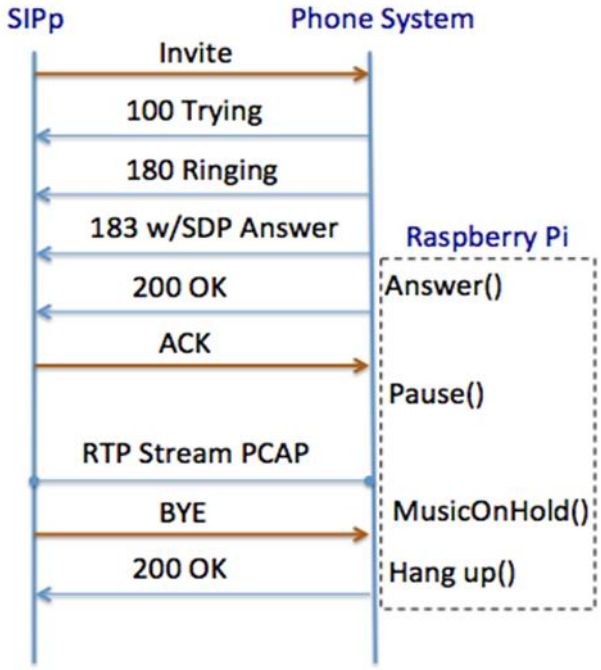

Figure 2. Call flow used to evaluate the performance of the IP telephony

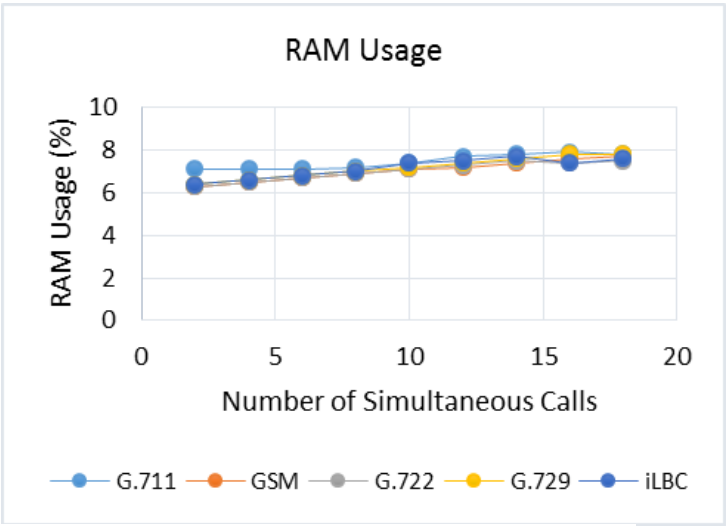

Figure 3. Percentage of RAM used during the test of the system, for different codecs and number of simultaneous callssystem based on the Raspberry Pi platform

Future work should include a deeper evaluation of the users' perceptions about the system, by scoring all the codecs already described and, perhaps, a more complex wireless environment. Since encoding/decoding processes are very CPU intensive, it would be interesting to assess the system also in front of a scheme where multiple codecs are used simultaneously. At first glance, a Raspberry Pi board would perfectly satisfy the needs of a small business in terms of voice communication although other coupled services such as videoconferencing still needs to be tested.

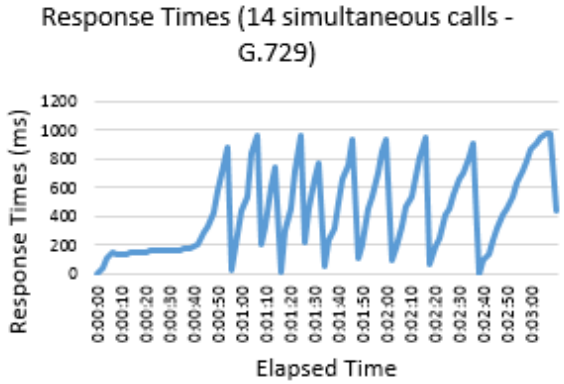

Figure 4. Comparison of response times for the different codecs and number of simultaneous calls

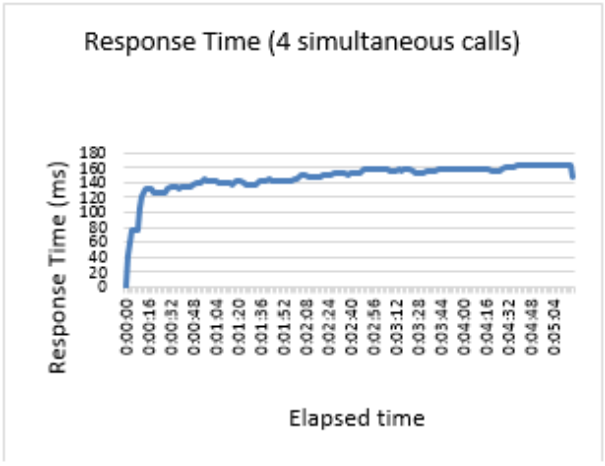

Figure 5. Failed calls per codec and number of simultaneous calls

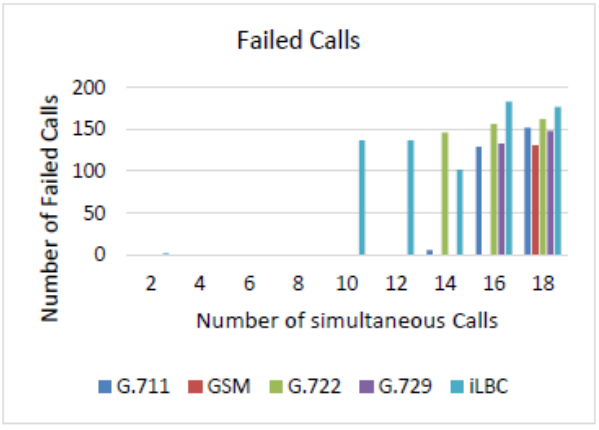

Figure 6. Failed calls per codec and number of simultaneous calls

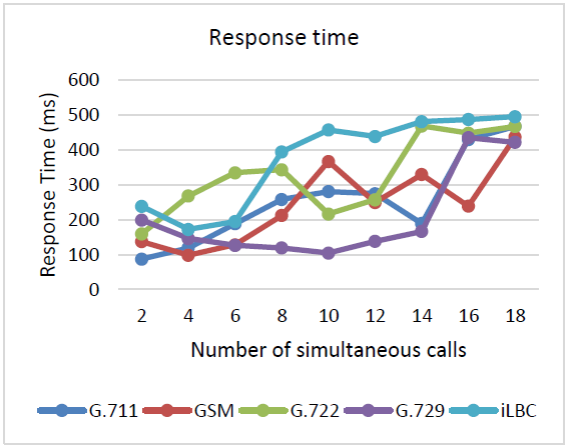

Figure 7. Response time varying – for 14 simultaneous calls and using G.729

REFERENCES

[1] B. Schaller & R. Stough (1996). The origin, nature, and implications of Moore's Law. PUBP801.

[2] Cisco, "Understanding Codecs: Complexity, Hardware Support, MOS, and Negotiation", [Available On Line] Retrieved from: http://www.cisco.com/c/en/us/support/docs/voice/h323/14069-codec-complexity.html#: June 2015.

[3] D. Mejía, & C. F. Garzón (2014). Implementación de un Prototipo de Sistema de Reportes Web para Telefonía IP. Revista Politécnica, 33(1).

[4] D. Peláez, & C. Tipantuña (2014). Servidor de comunicaciones unificadas con Raspberry Pi y Micro-Elastix.

[5] E. Landívar. (2008). Comunicaciones unificadas con Elastix. Páginas (15, 16).

[6] E. Upton, & G. Halfacree (2014). Raspberry Pi user guide. John Wiley&Sons.

[7] htop - an interactive process viewer for Linux, [Available On Line] Retrieved from: http://hisham.hm/htop/. Last visited: June 2015.

[8] ITU-T Rec. P.800, "Methods for subjective determination of transmission quality", International Telecommunication Union, Geneva, Switzerland (1996 Aug.).

[9] J. Van Meggelen, L. Madsen, & J. Smith (2007). Asterisk: the future of telephony. " O'Reilly Media, Inc.".

[10] M. Ahmed, & A. M. Mansor (2008, April). CPU dimensioning on performance of Asterisk VoIP PBX. In Proceedings of the 11th communications and networking simulation symposium (pp. 139-146). ACM

[11] M. N. Ismail (2010). Analysis of Secure Real Time Transport Protocol on VoIP over Wireless LAN in Campus Environment. International Journal on Computer Science and Engineering (IJCSE), 2(02), 898-902.

[12] O. Salcedo, D. López, & C. Hernández (2012). A comparative study of bandwidth usage running protocols SIP and IAX. Tecnura, 16(34), 171-187.

[13] Source Forge, Sipp Documentation. [Available On Line] Retrieved from: http://sipp.sourceforge.net/doc/reference.html. Last visited: June 2015.