SIP Infrastructure Performance Testing

MIROSLAV VOZNAK, JAN ROZHON
Department of Telecommunications
VSB – Technical University of Ostrava
17. listopadu 15, Ostrava
CZECH REPUBLIC
miroslav.voznak@vsb.cz, jan.rozhon@vsb.cz

Abstract: - This paper deals with a testing method suitable for SIP infrastructure. The performance testing is an issue of research and no standardized methodology has been adopted yet. We present the main ideas of our method that have been verified on open source software PBX Asterisk, as a representative of the well-known SIP solution. To be able to compare the machine/platform irrespective of a particular HW or SW version, we relate the results of the performance ratio with transcoding to performance without codec translation. This way we are able to achieve a comparative ratio that is independent of hardware.

Key-Words: - Asterisk, B2BUA, SIP Proxy, codec translation, performance testing

1 Introduction

The whole topic of SIP infrastructure performance testing is under development and there are no unified recommendations as how to perform the tests and what to pay attention to. Moreover, the proprietary solutions offer huge comprehensibility of testing scenarios but they do not use generally recognized means and ways to perform the testing, so the results may not be compatible. Many issues in this area have been solved by Transnexus. Their white papers [1] and general approach to the testing has significantly inspired our research because it is based on open source solutions and allows us to integrate basic thoughts mentioned in the IETF draft [2]. This RFC draft focuses on methodology for benchmarking SIP environment. Considering this information, it is obvious that there is a big gap in the area of SIP infrastructure performance testing and benchmarking. This gap and its elimination is the main motivation for our research. Simple SIP infrastructure performance testing configured in B2BUA mode and the examples of the output results are the main contribution of this paper.

2 State of the art

As mentioned in the introduction, there are some proprietary solutions for SIP testing, the main advantage of which is a huge comprehensibility of testing scenarios. However, in the real world, there are also disadvantages, such as high price and possible incompatibility of the results, as each company focuses on a different main area of interest. On the other hand, the IETF has published several drafts which have the methodology and the metrics of SIP infrastructure testing as their main topic of concern, see [2], [3] and [4]. These drafts try to define the basic terms for SIP benchmarking as well as the times, the measuring of which is important to gain the relevant results. Given the early stage of development of these drafts, there are no software or hardware means for SIP benchmarking that would utilize these drafts yet. Halfway to creating a suitable and generally applicable testing method is the Transnexus' SIP benchmarking model which can serve as an inspiration [1]. This company created a useful SIP infrastructure benchmarking method using an open source traffic generator SIPp. In order to develop a method which would reflect the main thoughts of the IETF drafts it is useful to modify the Transnexus' procedure and the results will be sufficient to determine the effectiveness of a system, the highest load which it can handle as well as the dynamically changing characteristics of a system.

3 Methodology

In order to perform SIP testing, we simulate both ends of the SIP dialogue to test the main part of the SIP infrastructure, the SIP server. The SIP server represents a set of servers always involving SIP Registrar and SIP Proxy or B2BUA (Back to Back User Agent). The latter is the most used solution in enterprise environment, for both SMEs (Small and Medium sized Enterprise) and LEs (Large Enterprise). Fig. 1 depicts its basic test hardware configuration.

This is a general configuration which does not reflect some hardware and software limitations; however it perfectly describes two essential elements of the testing. The first one is a special computer to perform the testing of RTP streams which allows us to use more sophisticated tools for capturing the network traffic without the RTP and SIP parts of the tests influencing each other. The second idea is that all the computers must be connected together just by a single switch. This allows the testing to be reproducible. Each measurement for every single codec translation case consists of several steps. Every single step takes 16 minutes, this means that for 15 minutes, 60-second long calls are going to be generated at a user-defined call rate. Then there is a 60-second period when the unfinished calls are going to be terminated. This repeats for every single step of the call rate. Every call consists of a standard SIP dialogue and RTP media. Because the load is not constant but increases slowly at the beginning of the test (first 60 seconds) and decreases at the end of it (last 60 seconds), the results taken after this starting period and before the ending one are the only ones which are going to be considered valid. The results are taken at two places: by means of the elements and the selected parameters. Fig. 2 shows the meaning of the RRD and SRD delays in more detail.

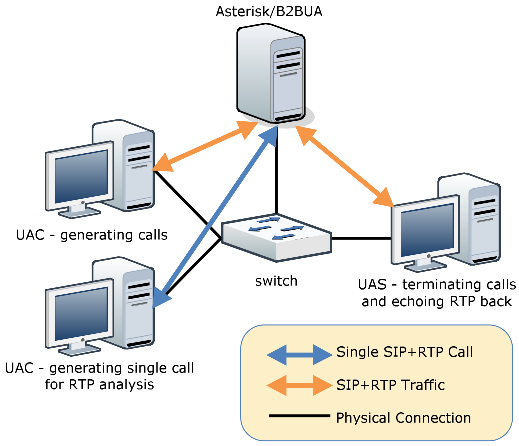

Fig. 1. Simplified testbed configuration.

A. *Elements*

- UAC: at this place, the number of (un)successful calls, durations of the message exchanges, call repartitions will be measured and RTP samples for analysis (on the separate UAC) will be captured.
- SIP server: at this place CPU, memory utilization and network traffic will be measured.

B. *Measured parameters*

- CPU Utilization.
- Number of (un)successful calls (just to know when to finish the test).
- Registration Request Delay (RRD), the time between first Register method and related 200 OK message [2].
- Session Request Delay (SRD), the time between first Invite method and related 180 Ringing message [2].
- Mean Jitter a Maximum RTP Packet Delay.

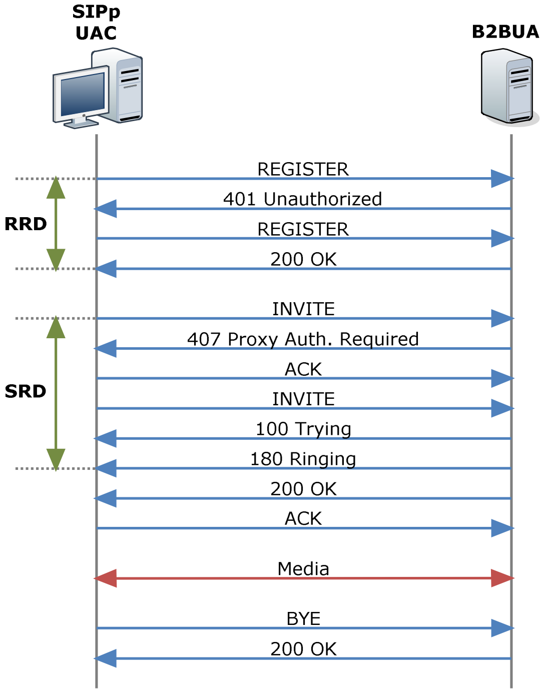

Fig. 2. Registration Request Delay and Session Request Delay in SIP Dialog.

Because we focus on testing effectiveness and speed of codec translation we were, at this point, able to determine the maximum load which the SIP server can handle from the SIP or RTP point of view. However, these results would only be valid for a single machine/platform and that is why we add one more step to the data analysis. The same procedure of testing as mentioned above is performed on a machine configured to allow media to only pass through the SIP server. The results taken during this test serve as a basis to which we relate all the other results. The relation is expressed in (1) as a performance ratio. The performance rating factor PRF is a ratio of a number of calls with codec translation PCT to overall performance P without transcoding.

$$P_{RF} = \frac{P_{CT}}{P} \cdot 100 \tag{1}$$

This step allows us to compare the results from hardware and platform independently.

4 Experiment

To simulate both UACs and UASs, we are going to use the SIP performance testing tool called SIPp [5]. This open source utility can simulate concurrent SIP calls. Moreover it allows measuring important times such as those defined in the IETF draft [2]. SIPp performs the calls which follow user-defined scenarios in xml language. This xml scenarios are distributed on every computer and the SIPp is invoked by using bash script and SSH. One of the computers works as a SSH client and controls the whole test by sending orders to other computers (SSH servers) via SSH. The message call flows exchanged between related UAC and UAS SIPp instances are depicted in the Fig. 3. The key values of hardware utilization on the B2BUA are measured by System Activity Reporter (SAR) every 10 seconds and 60 times, i.e. during the middle 10 minutes of the test when the number of simultaneous calls is constant. The media consists of a 60-second long music song recorded in G711u pcap file which is used by UAC. UASs are configured to use G711u-law, G711A-law, G726-32 and GSM codecs. The Asterisk PbX performs the codec translation. RTP streams can be captured and analyzed with Wireshark. Wireshark offers very complex means for RTP analysis [6]. However, the generation of RTP streams on the client side consumes a lot of CPU power, this means that we have to limit the number of calls generated by a single machine, which leads us to multiply the number of PCs running the UAC scheme.

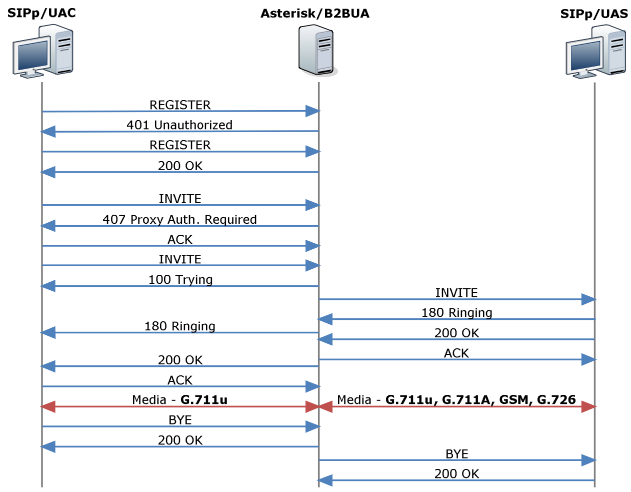

Fig. 3. SIP message flow exchanged through B2BUA.

The total number of the computers can be decided according to an estimated maximum load on a SIP server. Since in our case the SIP server is a PC with merely a dual-core processor, the total number of simultaneous calls will not exceed one thousand [7]. Each PC with our hardware configuration can generate around 200 calls. This is why the number of clients should equal or exceed four. In our case, four is just enough to perform the test. Servers can handle double load, and this is why there will be just two of them.

A. *Hardware and software configuration of UACs*

- Intel Celeron D 3,33 GHz, 1GB DDR
- Ubuntu 9.04 x64

B. *Hardware and software configuration of UASs*

- Intel Celeron D 3,33 GHz, 1 GB DDR
- Ubuntu 9.04 x64

C. *Hardware and software configuration of B2BUA*

- CPU – AMD Athlon 64 X2 5200+
- RAM – 4GB DDR2 (3,5 GB used due to x86 system)
- Debian 5.0 x86
- Asterisk PbX v. 1.6.2

The devices are connected to a Cisco C2960 switch and all PCs to the fast Ethernet ports. This is enough because the traffic load is distributed but B2BUA uses the Gigabit port of the switch. The topology of the network is depicted on Fig. 4.

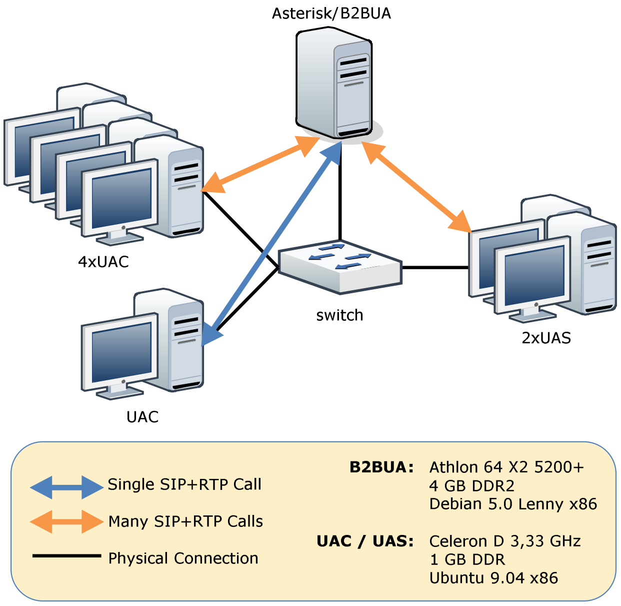

Fig. 4. Topology of the testbed in use.

The entire process of performance testing needs multiple computers to generate SIP traffic. To be able to successfully perform a test, the whole process must be automated. Therefore all the computers are being given orders by a Main UAC via SSH, we have created a set of bash scripts. On the Main UAC the bash script is invoked to deal with this task. In the first step, main UAC counts the number of calls that each computer should generate per one second period. Then it orders the UASs to register and start listening on UDP port 5060. This is done by a bash script. Secondly, SIPp on all UACs is invoked to generate traffic. As the last step, sar is invoked. This is done after 2,5 minutes to ensure the stable load has been reached already. The results contain CPU, memory and network statistics, and are stored in a file data_callrate.sar in binary format.

5 Results

For each category, there are two different charts. The first one shows the results for the case without codec translation and is colored in blue. The second shows the normalized values of the cases with a codec translation and is colored in three different colors. The first chart shows a simple relation between the number of concurrent calls passing through the B2BUA and its CPU utilization. The peak associated with 300 calls is probably caused by SAR which measures the data periodically. Thus it is possible that it took the samples with lower CPU utilization function which is not a constant but it is changing periodically. The second chart shows that (as expected) codec translation from G711u to G711A consumes about 20% more CPU power than a simple G711u case without translation. On the other hand, the most demanding is the G726-32bit codec. The lowest load returns the most interesting information. With the load of 60 calls, the differences in CPU power consumption for GSM and G726 is the highest compared to the one without codec translation. With higher loads it starts decreasing rapidly.

Mean CPU Utilization

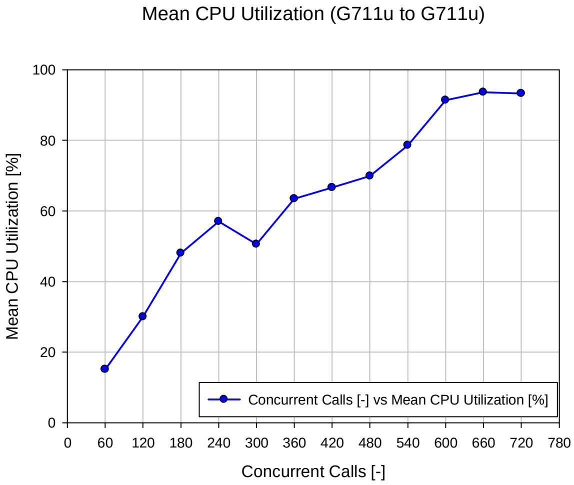

Fig. 5. Mean CPU utilization without transcoding

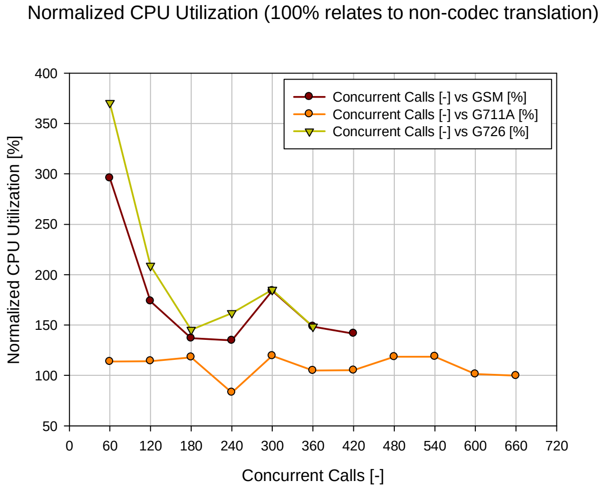

Fig. 6. Mean CPU utilization and its related normalized values.

RRD and SRD delays

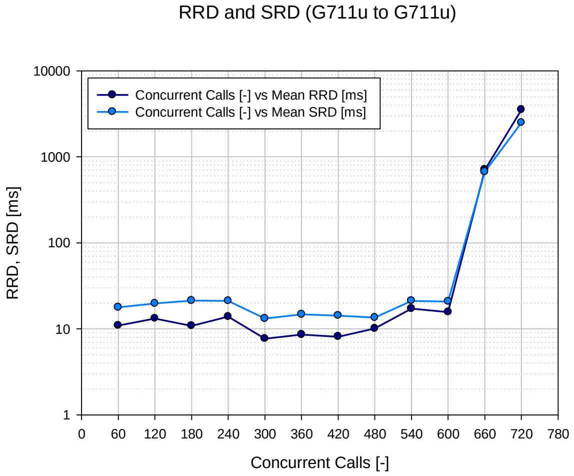

Fig. 7. RRD and SRD times.

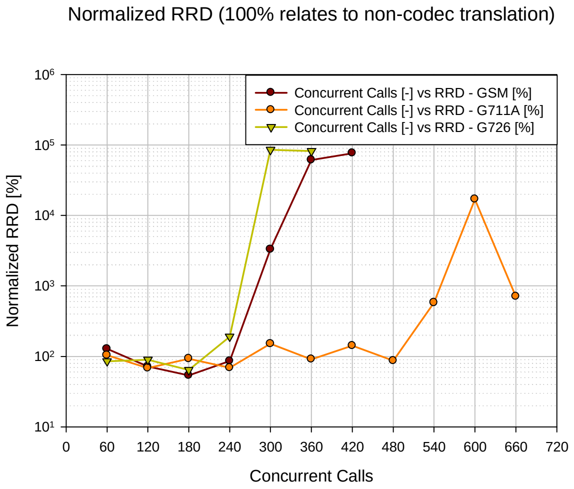

Fig. 8. RRD and its related normalized values.

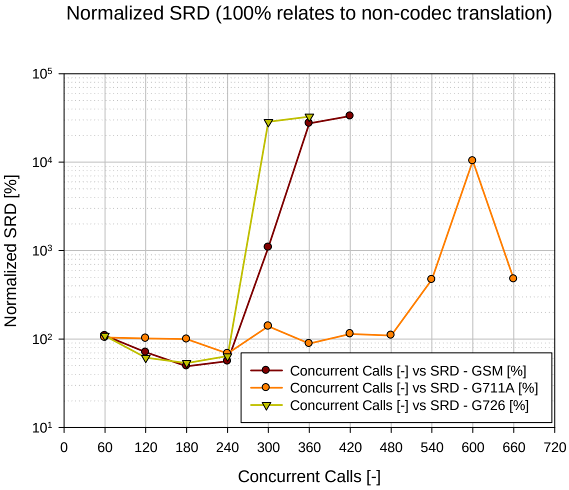

Fig. 9. SRD and its related normalized values.

Mean Jitter and Maximum RTP Packet Delay

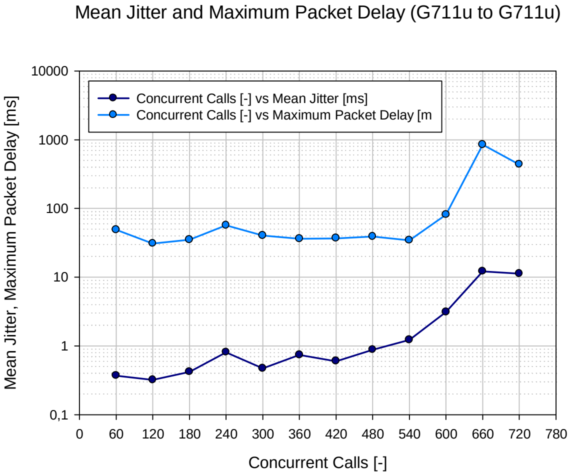

Fig. 10. Mean Jitter and Maximum RTP Packet Delay.

Charts on Fig. 8 and 9 clearly illustrate that the call is set up even quicker when there is a codec translation in use and the load is under 240 simultaneous calls. Then, as the CPU utilization increases, the delays get very long. The last G711A value for both charts is so low due to a rapid increase of delays for G711u to G711u case between 600-660 simultaneous calls.

Normalized values of mean jitter and maximum packet delay confirmed expected outcome as the values related to a small load are very similar to the main values from the case without codec translation. A very rapid decrease of both normalized values for G711A is caused by the increase of the main values from non-translation case and by the significant number of unsuccessful calls in this scenario.

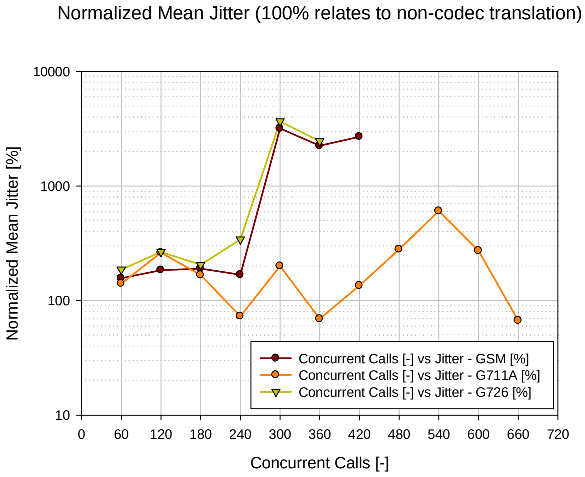

Fig. 11. Mean Jitter and its related normalized values.

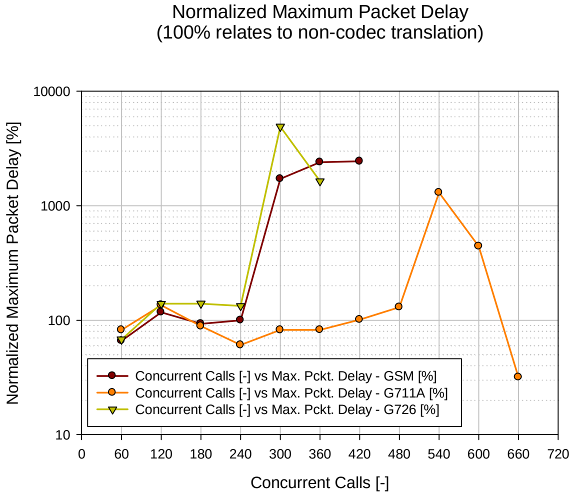

Fig. 12. Maximum RTP Packet Delay and its related normalized values.

6 Conclusion

The method of SIP infrastructure testing and benchmarking we present in this paper is designed for benchmarking SIP based VoIP infrastructure. It allows to determine the maximum load of the system, shows the dynamically changing characteristics of the system such as response times and packet delay. It is useful to decide which system should be installed in a particular environment.

Our designed method could be used in the INDECT project where a set of SIP servers will be operated. This benchmarking test is able to ascertain the performance limitation of designed SIP infrastructure.

Acknowledgement

This research has been supported by the “Optical Network of National Research and Its New Applications” (MSM 6383917201) research intent of the Ministry of Education of the Czech Republic.

References:

[1] Transnexus, *Performance Test of Asterisk V1.4 as a Back to Back User Agent (B2BUA)*, http://www.transnexus.com/White%20Papers/Performance_Test_of_Asterisk_v1-4.htm

[2] Malas, D., Morton, A. *SIP End-to-End Performance Metrics*, http://tools.ietf.org/-html/draft-ietf-pmol-sip-perf-metrics-04.

[3] Poretsky, S., Gurbani, V., Davids, C., *Terminology for Benchmarking Session Initiation Protocol (SIP) Networking Devices*, http://tools.ietf.org/html/draft-ietf-bmwg-sip-bench-term-01.

[4] Poretsky, S., Gurbani, V., Davids, C., *Methodology for Benchmarking SIP Networking Devices*, http://tools.ietf.org/html/draft-ietf-bmwg-sip-bench-meth-01.

[5] *SIPp, open-source project*, http://sipp.sourceforge.net.

[6] Wireshark Wiki, *RTP stream Analysis*, http://-wiki.wireshark.org/RTP statistics.

[7] *Asterisk Dimensioning*, http://www.voip-info.org/-wiki/view/Asterisk+dimensioning

[8] Voznak, M., *Voice over IP*. VSB-Technical University of Ostrava:, 2008

[9] *Asterisk – The open source telephony project*, http://www.asterisk.org/

[10] Meggelen, J., Smith, J., Madsen, L., *Asterisk: The Future of Telephony*. O'Reilly, 2007