Noname manuscript No.
(will be inserted by the editor)

Towards Automated Test and Validation of SIP Solutions

David Gonçalves · António Amaral · António Costa · Pedro Sousa

Received: date / Accepted: date

Abstract IP networks are currently the major communication infrastructure used by an increasing number of applications and heterogeneous services, including voice services. In this context, the Session Initiation Protocol (SIP) is a signaling protocol widely used for controlling multimedia communication sessions such as voice or video calls over IP networks, thus performing vital functions in an extensive set of public and enterprise solutions. However, the SIP protocol dissemination also entails some challenges, such as the complexity associated with the testing/validation processes of IMS/SIP networks. As a consequence, manual IMS/SIP testing solutions are inherently costly and time consuming tasks, being crucial to develop automated approaches in this specific area.

In this perspective, this article presents an experimental approach for automated testing/validation of SIP scenarios in IMS networks. For that purpose, an automation framework is proposed allowing to replicate the configuration of SIP equipment from the production network and submit such equipment to a battery of tests in the testing network. The proposed solution allows to drastically reduce the test and validation times when compared with traditional manual approaches, also allowing to enhance testing reliability and coverage. The automation framework comprises of some freely available tools which are conveniently integrated with other specific modules implemented within the context of this work.

In order to illustrate the advantages of the proposed automated framework, a real case study taken from a PT Inovação customer is presented comparing the time required to perform a manual SIP testing approach with the one time required when using the proposed automated framework. The presented results clearly corroborate the advantages of using the presented framework.

Keywords SIP · Automated Testing · Automated Validation · IMS Networks · VoIP · Telecommunication protocols

David Gonçalves
PT Comunicações, Portugal
E-mail: david-a-goncalves@telecom.pt

António Amaral
PT Inovação, Portugal
E-mail: antonio-mn-amaral@telecom.pt

António Costa
Centro Algoritmi, Department of Informatics
University of Minho, Portugal
E-mail: costa@di.uminho.pt

Pedro Sousa (Corresponding Author)
Centro Algoritmi, Department of Informatics
University of Minho, Portugal
E-mail: pns@di.uminho.pt

1 Introduction

The proliferation of IP networks has changed the way people interact and communicate. The impact of this technology is felt in both end users and service providers, having also consequences in the revenues of telecommunications networks [1]. Communication solutions for voice using IP networks are usually denominated by Voice over IP (VoIP) [2], being capable of converting analog voice into digital signals which are transmitted over packet switched networks. Among many of the the advantages of using VoIP, it is commonly accepted that users, companies and operators can take advantage from lower operational costs of VoIP technology, comparatively with other traditional telephone network approaches.

VoIP solutions often resort to specific protocols for media transmission (e.g. Real-Time Transport Protocol (RTP) [3]), and for session management (e.g. Session Initiation Protocol (SIP)). The SIP protocol [4][5] is nowadays widely accepted as a relevant signaling approach, being used for controlling multimedia communication sessions, as voice or video calls over IP converged networks. In fact, the SIP protocol is currently adopted by relevant architectural frameworks for delivering IP multimedia services, such as the IP Multimedia Core Network Subsystem (IMS) [6].

Meanwhile, the market for consumer SIP devices also continues to expand. In fact there are a lot of SIP related devices such as SIP Terminal Adapters, SIP Gateways, and SIP Trunking services providing replacements for ISDN telephone lines. Furthermore, many VoIP phone companies allow customers to use their own SIP devices, such as SIP-capable telephone sets and softphones. There is also a wide set of advanced applications that may also resort to SIP enabled devices. As a simple example, SIP-enabled video surveillance cameras can make calls to alert the owner or operator that an event has occurred, e.g. to notify that motion has been detected out-of-hours in a protected area. The SIP protocol can also be used in audio over IP for broadcasting applications where it provides an interoperable means for audio interfaces from different manufacturers to make connections with one another.

Thus, due to its widespread use and relevance in the telecommunication market, the SIP protocol has been the subject of several research works focusing on distinct aspects of the protocol behavior. As example, the work presented in [16, 21, 22] proposes a methodology allowing administrators to precisely measure the SIP Server performance and compare it to other software and hardware platform. Such studies also discuss the main ideas of a methodology allowing for testing the keystone of SIP based infrastructure, i.e. the SIP Server in both SIP Proxy and B2BUA (Back to Back User Agent) configurations. The performance issues related with the SIP protocol have also been addressed in other works, such as the presented in [17, 19]. Here, the authors present a novel logic-based approach to test the protocol performance through real execution traces and formally specified properties. In order to evaluate and assess the proposed methodology, the authors have developed a prototype and present experiments with a set of IMS/SIP properties. Also in the area of SIP testing, [24] presents a test system based on the notion of XML Protocol Templates of SIP call flows. These templates can be pattern matched against incoming messages and augmented with general purpose code to implement specific protocol responses.

The security aspects of the SIP protocol have also been the subject of several studies. In [18] the authors show that the robustness and reliability of generic SIP servers could be inadequate. For that purpose the authors have used a customized analysis tool with the ability to synthesize and launch different types of attacks. The work in [20] analyzes several attacks that are being made on the SIP architecture. Within such purpose, the authors give a brief introduction to some of the most common attacks on SIP, and then describe a framework to effectively test several security aspects of a SIP network and thereby help mitigate such attacks. The work present in [23] explores the known VoIP-related vulnerabilities and tests several of the more popular open source and commercial VoIP security tools with the intention of demonstrating the gap that exists between vulnerability and detection.

In another perspective, it is worth to mention the importance that test and validation processes might have in the deployment or upgrade of IMS/SIP network infrastructures. In fact, and merely as an illustrative example, SIP testing teams have many time to validate hardware or software changes in the network equipment assuring the correct behavior of the IMS/SIP infrastructure after the changes made. In general terms, the SIP protocol operates following a client-server model to exchange messages between the entities and SIP messages are usually presented in plain text, allowing a simple interpretation from the end users. Although the simple format of SIP messages apparently favors manual validation processes, the high diversity of SIP messages and the technological heterogeneity of common SIP networks turn manual validation approaches into an extremely time-consuming and prone to error processes. Thus, manual test and validation approaches may significantly delay the deployment or upgrade of SIP solutions in production networks, being necessary to devise automated solutions in this area. Similarly to other areas, automated testing implies a saving of resources making test processes more expedite, efficient and reliable.

In this context, this work focuses on devising efficient methods and tools allowing to improve SIP test and validation processes. In particular, this works intends to present a contribution for this research area by addressing the following topics in an integrated way: *i)* we firstly explain and discuss the main topics and challenges around the general problematic of SIP testing automation; *ii)* following that, a simple and efficient automated test and validation framework is described being able to speedup crucial tasks usually performed by IMS/SIP testing teams and finally, *iii)* using the implemented framework, and based on a real case study, illustrative results are presented illustrating the advantages of the proposed automated solution.

The remainder of the document is organized as follows. Section 2 discusses general aspects of the SIP testing problematic, also presenting a methodology to attain the pursued objectives. Section 3 describes the proposed framework, highlighting the major modules and tools integrating the devised solution. Section 4 illustrates the advantages of using automated SIP test and validation approaches, presenting the improvements obtained in a real case study. Finally, Section 5 presents the conclusions of the work.

2 The Problematic of SIP Testing Automation

Generally, the quality of a project depends on the ability to test and validate all the considered requirements. However, the correct and efficient mapping between requirements and test cases and their full implementation might be time consuming and difficult to reach. SIP testing processes act on a heterogeneous set of devices direct or indirectly affecting the flow of SIP messages in the network, sometimes requiring that each device be validated in a particular way. Besides the large number of validations to perform on different equipment there are other obstacles, such as the multiplicity of messages processed by SIP devices, the dynamism of certain message header fields, and different interpretation that some messages may have according with their precise location within a specific SIP flow. At a structural level SIP messages follow the structure defined in [7] with a syntax similar to that of HTTP. The main focus of SIP testing processes is the validation of the values associated with the message headers. However, the number of existing headers is quite extensive (even only considering the base RFC), being far superior when taking into account different RFCs of possible SIP extensions. Thus, the general question of the extensibility of SIP messages [8] is identified as a relevant problem within the SIP test and validation problematic.

Typically, signaling protocols are many times understood as having low impact on the network performance. However, in some scenarios, SIP messages can have huge sizes, with relevant impact on validation processes. Additionally, in SIP scenarios the validation of a particular requirement is rarely achieved only with the analysis of a single header, being necessary to analyze for each requirement a set of distinct operational behaviors along with the corresponding values observed in specific message headers. This makes validation tasks complex or at least extended (if dividing the overall test in lower complexity individual tasks). Additionally to the extensibility of SIP messages, the associated headers also have high dynamism. Unlike the vast majority of other protocols, SIP headers fields do not show restricted values, due to the fact that the allowed values are defined by an Augmented Backus-Naur Form (ABNF) grammar [9]. Thus, a large diversity of header field values might be observed in real SIP networks, causing difficulties comparatively with other scenarios where validations are only focused on boolean or enumerated fields.

As regards to SIP message flows, it is also worth to mention the need to validate SIP messages in view of where they were collected in the network, as similar messages may have different functions depending on their location. It is also commonly observed in real IMS/SIP networks that different devices with similar functions may present slightly distinct behavior when submitted to the same test, e.g. by adding different messages headers or handling messages in distinct way. Sometimes, these nuances exist even in correctly standardized and homologated devices. This relates to the fact that some normalization directives do not fit all aspects of the SIP protocol, giving some freedom to equipment manufacturers of having different methods of operation.

All the above discussed SIP related topics highlight both the inherent complexity of testing IMS/SIP infrastructures and the importance of devising automated testing tools in this area. The next sections will further analyze and discuss other related questions such as the distinct types of testing objectives (section 2.1) and a methodology for the test and validation of SIP scenarios (section2.2).

2.1 Testing Objectives

The tests are an integral part of all projects, allowing to validate if the operation of a given system meets the specified requirements. Even considering that usually the development life-cycle of any system includes a test phase, it is necessary to perform tests on different phases and with distinct objectives. Tests can be of distinct types and sometimes distinct nomenclatures exist to classify them. Nevertheless, some illustrative examples are mentioned in Table 1.

In the practical case of IMS/SIP scenarios, each solution is usually validated on distinct occasions. When the solution is developed the initial test target is to validate if the observed behavior of the system is the expected behavior, which is usually designated by functional testing. Before deploying the solution, a limited subset of tests is performed in order to verify that the system behaves in accordance with the expected (sanity

Table 1 Some Examples of Testing Objectives

Test Type	General Description
Compatibility Testing	Compatibility tests are non-functional tests, conducted in order to validate the behavior of the solution in an environment different from the one used during the development. (e.g. validate an application on different operating system versions).
Sanity Testing	Correspond to rapid tests performed in order to validate if the solution operates correctly. These are constituted by a small number of tests, with the primary objective to validate if the solution is mature enough to experience a wider range of tests.
Regression Testing	Aims to find differences in the operation of a system comparatively with previous versions of the system. Typically, regression tests are applied to validate hardware or software changes by checking if the system behavior is still the expected one.
Acceptance Testing	Tests performed by the customer after validation by the company that developed the system. Acceptance tests usually cover all the developed use cases, aiming to test the solution in a scenario as close as possible similar to the customer environment.
Performance Testing	Verify the behavior of the system under distinct load situations, also validating the stability, scalability and fault tolerance capabilities of the implemented solution.

testing). Then, it is necessary to validate the solution as a whole on an environment as close as possible similar to the production network (acceptance testing). Furthermore, even after the delivery of a given solution, it is necessary to repeat validation tests whenever there are changes in the software or hardware components of the system (regression testing). Regression testing is also an important asset to be used when the equipment to validate interacts with other devices in the network infrastructure. On such specific case, it is then necessary to ensure that the traffic inputs/outputs exchanged between different devices are identical to the ones observed in previous versions of the system.

A main objective of an automated test and validation approach is to achieve lower test execution times. This is justified by the amount of tests that are usually required to be made on a given infrastructure. As example, based on our previous experience in IMS/SIP equipment, a battery of tests can easily reach three hundred tests cases per access. Assuming a real illustrative case where an equipment has three different accesses (e.g. GPON, ADSL and a dedicated VPN access) this performs a total of nine hundred tests. In the case that each test takes on average one minute to be completed, then fifteen hours of consecutive testing operations are required. Nevertheless, an automated testing approach will not only speed up testing tasks, but will also improve the reliability of such critical procedures.

2.2 A Methodology for Test and Validation of SIP Solutions

The testing framework here proposed intends to be an effective asset to automatically submit a specific SIP equipment to a battery of tests. As example, in the specific case of regression testing, the framework can be used to test if after a hardware or firmware update a given equipment stills shows the expected input/output SIP signaling dynamics. The test process is expected to take place in a controlled environment (here denominated as testing network) where the testing team has full control of the equipment, also submitting the required SIP traffic and observing the corresponding equipment output behavior. Thus, for validation purposes, SIP equipment will be submitted to a battery of tests where, for a given set of inputs, the corresponding generated outputs will be validated. Moreover, the tested scenarios should also reproduce the SIP interactions observed in the production network, thus assuring that the equipment will preserve the expected behavior when deployed there.

An illustrative example of an architecture of an IMS network is shown in Figure 1, integrating distinct types of SIP equipment. Typically, such architectures are many times constituted by equipment from different vendors, having also distinct support teams for each device. For

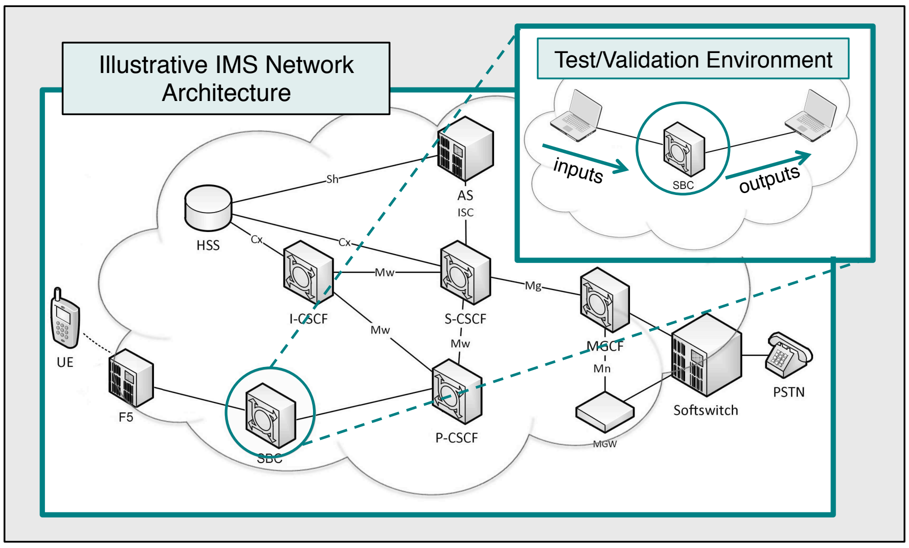


Fig. 1 Example of a typical IMS architecture (illustrating a production network from a PT Inovação customer), where the Session Border Controller (SBC) equipment will be submitted to a test/validation process.

instance, when aiming to test an SBC[1] (Session Border Controller) [10] equipment in the same conditions of Figure 1, it would be necessary to control the inputs coming from the F5 system (in this case a F5 BIP-IP load balancing equipment), as well as the outputs sent to the P-CSCF equipment. In this perspective, an automated test and validation framework should be able to replicate and integrate a given SIP equipment in the testing network, also requiring versatile tools to freely generate the SIP traffic to be used during the tests. Such traffic generation tools must be conveniently configured to replicate all the SIP signaling dynamics observed in the production network, also integrating additional standard test cases, if required.

In order to pursue an automation framework for SIP test and validation three distinct but complementary processes are identified as essential: *i)* test scenario preparation, *ii)* test cases creation and *iii)* tests execution. The test scenario preparation process performs a set of configuration tasks conveniently configuring and replicating the selected equipment to the testing network, also coordinating the integration of such equipment with other existing modules of the testing network. The test cases creation process is responsible for mapping a given set of system requirements into a set of test cases used to validate the equipment. This process may also take advantage of preliminary captures of SIP traffic that were made in the production network (e.g. input/output traffic received/sent by a specific SIP equipment). Finally, the test execution process will automatically execute a battery of test cases, from which a detailed validation report will be generated to the testing team members.

[1] A SBC is a device regularly deployed in VoIP networks to exert control over the signaling and the media streams involved in setting up, conducting, and tearing down telephone calls. The SBC enforces security, quality of service and admission control mechanism over the VoIP sessions.

3 A Framework for Scenario Replication and Test Automation of SIP Environments

This section describes in detail the devised solution for automated test/validation of SIP solutions. Figure 2 presents in a simplified perspective the implemented automation framework, highlighting the main modules included and their corresponding interactions. The objective was to devise a simpler but effective platform, taking also advantage of possible freely available tools. In this case, and as observed in Figure 2, the framework integrates the SIPp tool and a modified version of the sniff2sipp tool. These tools were integrated with other additional modules specifically devised in this work to add complementary automated functionalities to the framework. Overall, the proposed automation framework allows to reduce test and validation times of SIP environments, while also enhancing testing reliability and coverage.

The following sections overview some aspects of the devised framework and associated modules. The mainly focused aspects are: the used tools for the emulation of SIP traffic (section 3.1), the test scenario preparation framework component (section 3.2), the test cases creation framework component (section 3.3) and the test execution component of the framework (section 3.4).

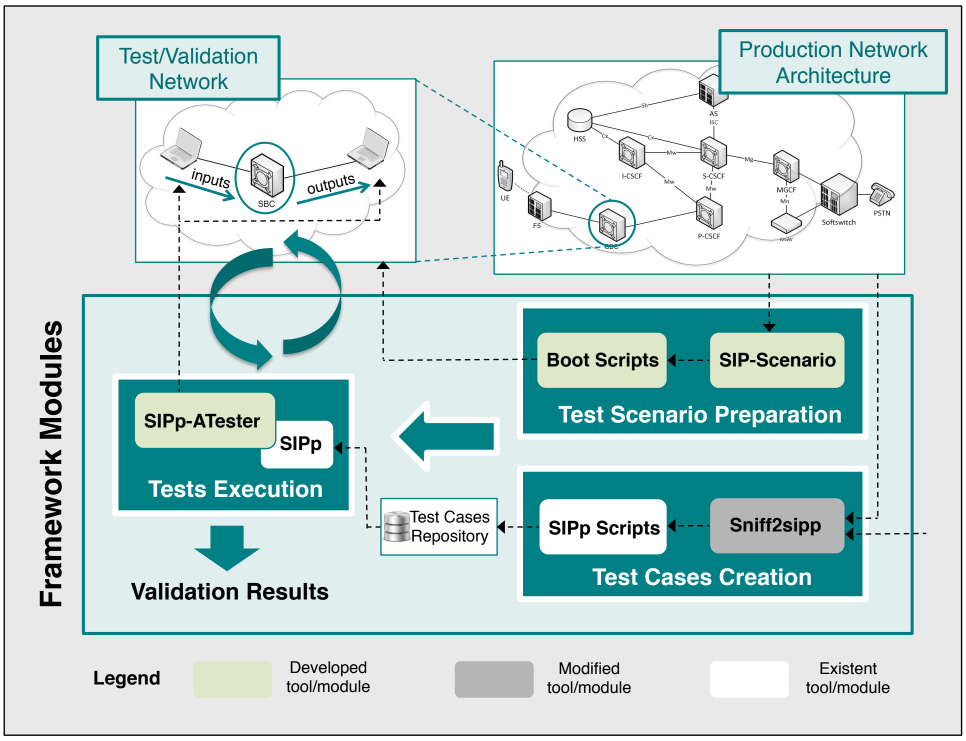


Fig. 2 Functional view of the modules integrating the proposed automation framework for test/validation of SIP scenarios.

3.1 Emulation of SIP Traffic

Emulation tools for SIP traffic generation are an important asset for testing platforms integrating real SIP equipment. To this end, several alternative tools were analyzed in this work context and the decision was the selection of the SIPp tool. A crucial argument for this choice was the great acceptance of SIPp by relevant entities in the IMS area, namely the Fraunhofer research organization [11].

The SIPp [12] is an open source tool for SIP traffic generation characterized by being easy to use to emulate sender/receiver SIP entities through XML scripting. SIPp is able to be used with the objective of injecting traffic in real SIP equipments, also enabling to perform specific validation processes over SIP flows. In this approach, the transmitting and receiving SIP entities are in fact represented by two XML files. These scripts will be executed by SIPp, indicating the SIP messages to be sent and which are expected to be received at the destination. Thus, the use of SIPp makes possible to emulate in the testing network a sender entity transmitting SIP traffic to a real SIP equipment, being also possible to analyze and validate the output which is subsequently sent to a given destination. Besides that, SIPp also allows the validation of SIP dynamics through the use of regular expressions, making it possible to check and validate specific headers in different SIP messages.

As depicted in Figure 2, the SIPp tool was conveniently combined with other modules of the framework. As example, the SIPp tool will receive as inputs the scripts generated by the test cases creation module, and will be used by the SIPp-ATester process (which is described later in section 3.4) during the test execution phase to trigger in the virtual machines the generation of SIP traffic.

3.2 Test Scenario Preparation

The test scenario preparation module (Figure 2) is responsible for ensuring that the equipment under test is ready to be tested by the SIPp-ATester application in the testing network. The testing approach followed is based on the assumption that it is not easy to fully replicate the production network in the testing environment. The number and diversity of hardware equipments that may be involved also discourages attempts to partially replicate a subset of the network. Equipments are therefore usually tested individually, whenever possible, ensuring however that they are operating under the exact same conditions found in the production network. Thus, the major goal of this module is to conveniently prepare each equipment to meet that requirement.

The preparation process is done in three well identified steps *i)* collect and validate configurations *ii)* compute the necessary reconfigurations and *iii)* update equipment configuration. The first and second steps are automated by an interactive application, called SIP-Scenario, that was developed from scratch for that purpose. Based on the report produced by the SIP-Scenario application, a set of configuration scripts (Boot Scripts in Figure 2) is then generated. Configuration update is done by running those scripts. The SIP-Scenario application and the Boot Scripts are now briefly described.

3.2.1 SIP-Scenario

The SIP-Scenario application was written in Java Language, mainly because of its well known write-once-run-everywhere characteristics. Since it has to deal with several distinct hardware equipments, with different operating systems, it was also built in a modular way. A specific module needs to be written and added to the application for each device that it supports. For each equipment a different method may be used to collect and analyze its configuration. For instance, for some equipments like the SBC (in Figure 1), the best approach is to produce a remote backup copy of its current configuration in a compressed XML file for complete offline analysis. The backup file is afterwards analyzed with an appropriate XML parser. For other equipments, like the BIG-IP products from F5 (F5 in Figure 1), the best way to collect and validate its configuration is to login and issue specific "show" commands using a SSH connection.

Whatever the method used to obtain them, all relevant configuration elements are inventoried with an indication whether or not its values should be modified for correct operation in the testing network. This results in a complete report with indication of configuration elements to be added, removed or updated. The knowledge needed for this operation depends on the equipment and comes from the experience acquired by experienced testing engineers. One typical mandatory first test is the validation of the basic network level configuration elements (interfaces, VLANs, IP addresses, IP routes, firewall rules) that may affect connectivity in the testing network. If these configurations are not suitable for usage in the testing network, they are sent to a support team for further analysis.

3.2.2 Boot Scripts

The configuration updates identified by SIP-Scenario application are applied using scripts written in the LUA [13] scripting language. The LUA language was selected because of its well known portability and performance characteristics. LUA scripts are executed with extra-Putty [14] because it supports SSH connections. All equipment specific commands are implemented in the form of supporting libraries. Those libraries provide a set of easy to use high level functions, that hide equipment dependent code from the LUA script. For instance, invoking *init()* and *restoreBackup(backup)* in sequence, applies a backup configuration. No specific knowledge of the equipment is therefore needed to prepare and run a LUA script. In addition, this also makes BootScripts more legible.

3.3 Test Cases Creation

The test cases creation module is designed to assist the production of SIP tests in the form of SIPp Scripts, that can be stored in a repository for later usage (i.e. the Test Cases Repository of Figure 2). SIPp scripts are created automatically using a modified version of an open source tool called Sniff2sipp [15] that was made in the context of this work. This tool, written in perl language, is able to receive captures of SIP traffic and produce a SIPp Script that can later be used to generate and validate a similar traffic pattern.

The capture of the required real world SIP traffic has to be done in the production network, with the equipment to be tested under normal operation, and according to test plans that were initially designed by the developing team. The traffic capture is first filtered off all irrelevant flows, and then used by the modified Sniff2sipp tool to generate the SIPp script. During parsing, SIP messages are transformed into templates, with specific values replaced either by variables or regular expressions. Variables can be instantiated at execution time for all SIP messages to send, while regular expressions can be evaluated to validate SIP messages received (see the Figure 3).

3.3.1 Sniff2sipp

The Sniff2sipp tool receives traffic captures as input and generates SIPp scripts as output. It can work directly with live captures or with files containing captured traffic. It is able to parse the traffic, extracting IP packets from underlying Ethernet frames. It was modified to further detect and accept the presence of the IEEE 802.1Q VLAN header. During parsing, Snif2sipp replaces specific field values by named variables to include in the resultant SIPp script. The identification of a remote party, for instance, is replaced during parsing by *[remote_ip]*. This allows for more flexibility in the ex-

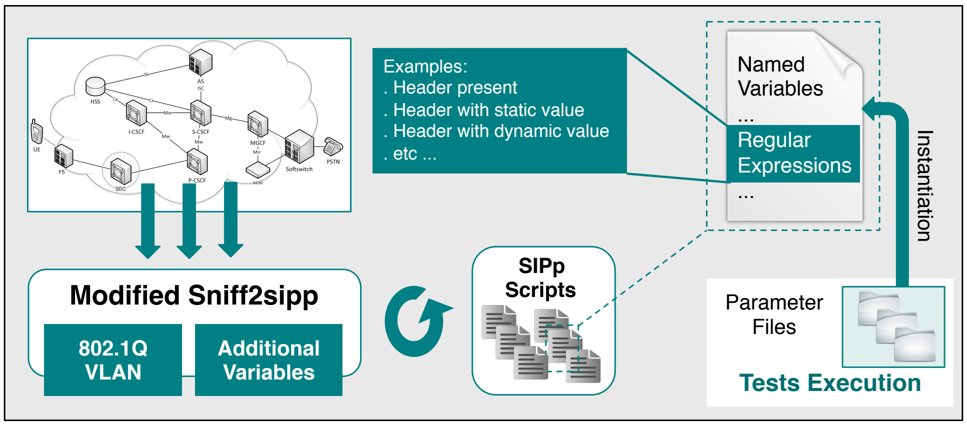


Fig. 3 A modified version of the Sniff2sipp tool used in the Tests Case Creation module of the framework.

ecution scenario. Sinf2sipp was also modified to include more of those useful variables (see Figure 3).

3.3.2 SIPp scripts

SIPp Scripts describe either SIP client or SIP server behavior. They are really XML files easy to parse and interpret, that describe the SIP messages to be sent and the SIP messages that are expected to be received. SIPp scripts can include static information that never changes in tests and dynamic information that has to be changed during tests, like incrementing fields, call ids, user ids, etc. Dynamic information is described by variable fields as mentioned before. During execution, SIPp variables have to be instanced from a parameter file passed as parameter.

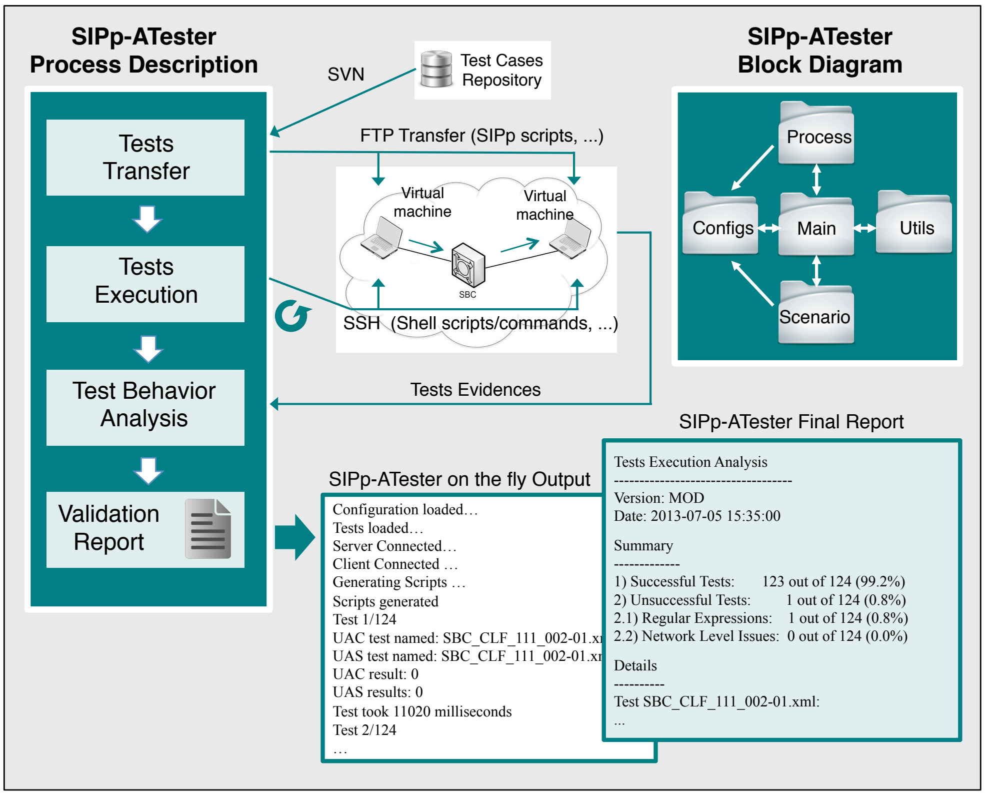
For output validation, SIPp scripts use powerful Perl regular expressions to detect expected patterns in the received messages. Regular expressions are of three different types (Figure 3): *i)* header present *ii)* header with static value *iii)* header with dynamic value. Regular expressions of type *i)* detect if a specific header is present in the headers fields, irrespectively of its value. Regular expressions of type *ii)* validate the presence of a specific header with a specific static value (e.g. RURI in emergency calls with the correct number). Regular expressions of type *iii)* detect the presence of a dynamic value in a header, that can change from test to test, but with the expected semantic (e.g. the user part in a field that should identify the user).

3.4 Tests Execution

The test execution module that integrates Figure 2 has a crucial relevance within the devised framework. This module is responsible to handle four distinct tasks, namely: tests transfer, tests execution, tests analysis and validation reports generation. A specific application, entitled SIPp-ATester, was developed in this work context aiming to attain such objectives. A high level functional view of the SIPp-ATester application is depicted in Figure 4.

As depicted in Figure 4 in the initial phase the SIPp-ATester starts by accessing the test files stored in a SVN repository. Based on the configured testing scenario, a group of SIPp scripts is automatically transferred (e.g via FTP) to the virtual machines in the testing network which are enabled to execute SIPp scripts. The SIPp-ATester is also responsible to inform the virtual machines about which tests should be executed and how to execute them, sending for that purpose additional configuration parameters to the testing network devices (e.g. connectivity information, timers, specific input script arguments, among others). Simultaneously with the tests execution taking place in the testing network, the SIPp-ATester also collects information regarding the test results, after which is possible to generate the validation reports. All these tasks are automatically performed and controlled via SSH connections established with the virtual machines. The connections are mainly used to control the execution of specific commands and applications in the virtual machines, and also to remotely trigger the generation of additional shell scripts controlling all the test execution phase, e.g. scripts controlling an ordered execution of several individual SIPp tests.

As also shown in Figure 4, the SIPp-ATester generates an intermediary output report through all the test execution phase, reporting for each individual test the obtained result and corresponding execution time. After the execution of all the individual test cases a final report is presented summarizing the percentage of successful/unsuccessful tests, also providing additional information regarding the failed cases. This informa-


SIPp-ATester Process Description
Tests Transfer
Tests Execution
Test Behavior Analysis
Validation Report
SVN
Test Cases Repository
FTP Transfer (SIPp scripts, ...)
Virtual machine
Virtual machine
SBC
SSH (Shell scripts/commands, ...)
Tests Evidences
SIPp-ATester Block Diagram
Process
Configs
Main
Utils
Scenario
SIPp-ATester on the fly Output
SIPp-ATester Final Report

```
Configuration loaded…
Tests loaded…
Server Connected…
Client Connected …
Generating Scripts …
Scripts generated
Test 1/124
UAC test named: SBC_CLF_111_002-01.xı
UAS test named: SBC_CLF_111_002-01.xı
UAC result: 0
UAS results: 0
Test took 11020 milliseconds
Test 2/124
…
```

```
Tests Execution Analysis
-----------------------------------
Version: MOD
Date: 2013-07-05 15:35:00

Summary
-------------
1) Successful Tests:       123 out of 124 (99.2%)
2) Unsuccessful Tests:       1 out of 124 (0.8%)
2.1) Regular Expressions:    1 out of 124 (0.8%)
2.2) Network Level Issues:  0 out of 124 (0.0%)

Details
----------
Test SBC_CLF_111_002-01.xml:
...
```

Fig. 4 A high level functional view of the SIPp-ATester application and associated application block diagram.

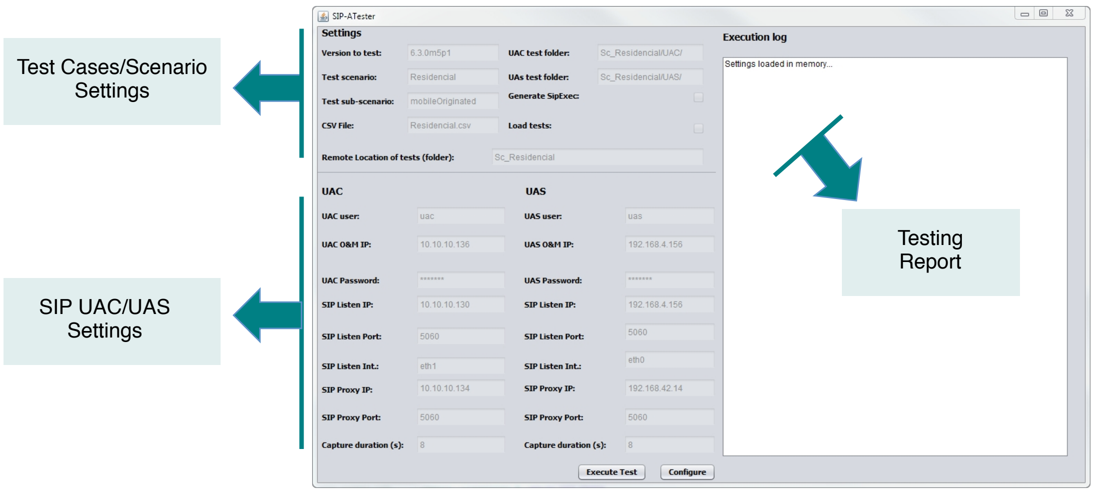
tion will allow a detailed analysis from the testing team about the tested SIP equipment. As presented in the upper right corner of Figure 4 the SIPp-ATester application was implemented in five modules, with the *Main* block controlling all the application dynamics. The *Configs* module reads a set of configuration files gathering essential configuration information about the virtual machines, the scenario to test and the tests to be executed. The *Process* module organizes the test files in the virtual machines being also responsible for generating shell scripts from which several SIPp scripts will be coordinately executed. The *Scenario* module provides the necessary input arguments to be used by the shell scripts generated by the Process module. Finally, the Utils block deals with the auxiliary tools used by the SIPp-ATester, such as FTP, SSH and other embedded communication clients.

In order to interact with the SIPp-Atester module it was decided to provide the user with two distinct alternatives. A command-line interface is available allowing to provide several input parameters, run the tests and collect the obtained results. The command-line interface is often preferred by more advanced users, as it provides a more concise and powerful mean to control the test execution phase. Furthermore, it also allows the use of scripting to implement additional user defined functionalities and further enrich the devised framework.

Alternatively, a graphical user interface is also available allowing the user interaction with the devised framework. In this case, Figure 5 shows an example of an interface that the user may use to interact with the SIPp-ATester application. The interface receives as inputs several configuration parameters that are needed for the test execution phase. The button *Execute Test* of the interface triggers the test execution phase, automatically transferring the SIPp scripts to the virtual machines in the testing network and also automatically controlling the tests execution via SSH connections. The results of the performed tests can be also observed in the execution window of the interface (right side of Figure 5).

4 Framework Applicability and Illustrative Results

Some of the inherent advantages of the proposed automation framework were already highlighted in previous sections of this work. The proposed solution allows to effectively reduce the SIP test and validation times while enhancing testing reliability and coverage, thus also indirectly contributing to speed up the in-



Settings

Version to test: 6.3.0m5p1
UAC test folder: Sc_Residencial/UAC/
Test scenario: Residencial
UAs test folder: Sc_Residencial/UAS/
Test sub-scenario: mobileOriginated
Generate SipExec: ☐
CSV File: Residencial.csv
Load tests: ☐
Remote Location of tests (folder): Sc_Residencial

UAC

UAC user: uac
UAC O&M IP: 10.10.10.136
UAC Password: *******
SIP Listen IP: 10.10.10.130
SIP Listen Port: 5060
SIP Listen Int.: eth1
SIP Proxy IP: 10.10.10.134
SIP Proxy Port: 5060
Capture duration (s): 8

UAS

UAS user: uas
UAS O&M IP: 192.168.4.156
UAS Password: *******
SIP Listen IP: 192.168.4.156
SIP Listen Port: 5060
SIP Listen Int.: eth0
SIP Proxy IP: 192.168.42.14
SIP Proxy Port: 5060
Capture duration (s): 8

Execution log

Settings loaded in memory...

Fig. 5 Example of a graphical interface used to interact with the SIPp-ATester module of the proposed framework.

stallation, upgrade or repair processes of IMS/SIP infrastructures. Furthermore, an additional advantage of the devised framework is that it requires lower level of IMS/SIP technical expertise, comparatively with manual SIP test and validation approaches.

To illustrate the advantages of the proposed solution, a real case study taken from a PT Inovação customer is presented. On this customer, during each year three to four firmware updates are performed, requiring the test of some SIP equipments. The objective is to compare the time required to perform the manual testing approach with the one required using the automation framework. Within this context three distinct testing scenarios are considered when presenting the illustrative test/validation results:

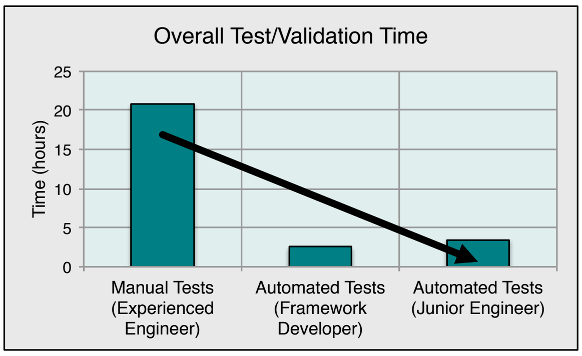
– Manual Tests (Experienced Engineer): a traditional manual test and validation approach performed by a senior engineer.
– Automated Tests (Framework Developer): a test and validation process using the automated solution performed by the framework developer, i.e. the responsible for the software development of the proposed testing framework.
– Automated Tests (Junior Engineer): a test and validation process using the automated solution performed by a junior engineer with limited experience in IMS/SIP networks.

For each one of the considered testing scenarios two distinct metrics will be analyzed, namely: *i)* the overall test/validation times and *ii)* the total number of test cases analyzed in the experiment. Figures 6 a) and b) summarize the results obtained from a set of SIP regression tests made in the SBC (Session Border Controller) equipment of such PT Inovação customer network.

The last manual SIP validation test made in such customer was performed by an experienced engineer in the area of IMS/SIP and SBC equipment. This manual validation lasted around 20 hours over which roughly 25 tests were remotely performed on the customer network (see the first columns of Figures 6 a) b)). This time encompasses tasks such as the terminal preparation, test execution, capturing evidences, filtering and flow analysis, analysis of headers and indication of the test results.

In contrast, when using the automated testing platform, the framework developer obtained a total test and validation time around 2.5 hours, but in this case performing a total of 300 test cases within such small period (see the second columns of Figures 6 a) b)). In this case study, for an overall test and validation time approximately ten times lower than the obtained by manual approaches, the capability of executing automated tests increased about ten times. Thus, when accounting the average time required to perform each individual test, the automated solution attained improvements around two orders of magnitude over the traditional manual testing approaches.

Beyond the advantage of allowing a fast execution and widening the coverage and reliability of SIP testing solutions, the proposed framework is also easy to use. In fact, for the same scenario, a junior engineer held automated validations using the proposed framework and obtained test/validation times very close to the ones obtained by the Framework Developer, for the same number of test cases executed (see the third columns of Figures 6 a) b)). In this way, the proposed automated platform allows to significantly simplify SIP testing processes, making possible that SIP test/validation tasks could be also performed by individuals not having high



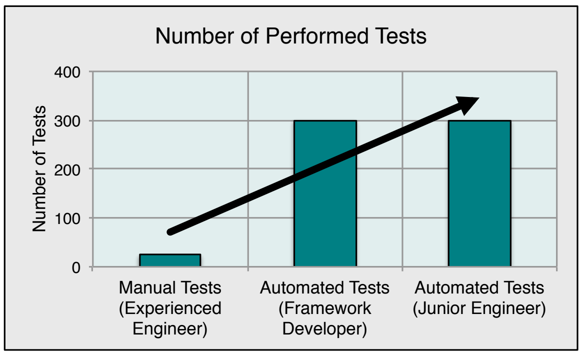


Fig. 6 Illustrative results obtained from SIP regression tests of an SBC equipment, comparing manual and automated test/validation approaches a) Overall test/validation time and b) Number of performed tests.

IMS/SIP technical expertise. In short, an overall analysis of the results of Figures 6 a) b) clearly corroborate the potential gains of using the proposed automated SIP testing/validation approach.

5 Conclusions

In this article we propose a framework enabling automated test and validation of SIP solutions in the context of IMS networks. After a preliminary discussion about the general problematic of SIP testing automation, three distinct processes were identified as critical to be integrated in an automation framework. Based on such assumptions, an experimental framework was devised and implemented integrating proprietary modules conveniently aligned with other already existent or modified available tools. The resulting automated testing framework is an important asset to test SIP equipment and can be used, for instance, by SIP testing teams to validate hardware or software changes in the IMS/SIP network infrastructure.

In order to illustrate the potential gains of the solution a real case study was presented where manual SIP testing approaches were compared with automated solutions. The tests were performed in a PT Inovação customer where during each year three to four firmware updates are performed, thus requiring the test of some SIP equipments. The presented results have shown that relevant improvements are attained in three distinct perspectives. Firstly, it is possible to significantly reduce the overall test and validation times. Secondly, and in addition to test time improvements, there is a clear increase in the number of test cases validated using the automation framework, which translates into enhancing testing reliability and coverage. Last, but not least, the proposed automated platform allows to simplify SIP testing processes, avoiding that such critical tasks could only be executed by individuals having very high IMS/SIP technical expertise.

Acknowledgments

This work has been supported by FCT - Fundação para a Ciência e Tecnologia within the Project Scope: PEst-OE/EEI/UI0319/2014.

This research work was developed within the collaboration of PT Inovação (`http://www.ptinovacao.pt/en/`).

References

1. Antonio Cuevas, J.I. Moreno, P. Vidales and H. Einsiedler. The IMS Service Platform: A Solution for Next-Generation Network Operators to be More than Bit Pipes. Communications Magazine, IEEE, 44(8):7581, 2006.
2. Jonathan Davidson, James F. Peters, Manoj Bhatia, Satish Kalidindi and Sudipto Mukherjee. Voice over IP Fundamentals, Cisco Press; 2 edition, 2006.
3. H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson. RTP: A Transport Protocol for Real-time Applications, RFC 3550, 2003.
4. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler. SIP: Session Initiation Protocol, RFC 3261, 2002.
5. Alan B. Johnston. SIP: Understanding the Session Initiation Protocol, Artech House; 3 edition, 2009.
6. Miikka Poikselka and Georg Mayer. The IMS: IP Multimedia Concepts and Services, 3rd Edition, Wiley, 2009.
7. P. Resnick. Internet Message Format, RFC 2822, 2001.
8. J. Rosenberg and H. Schulzrinne. Guidelines for Authors of Extensions to the Session Initiation Protocol (SIP), RFC 4485, 2006.
9. D. Crocker and P. Overell. Augmented BNF for Syntax Specifications: ABNF, RFC 5234, 2008.
10. Metaswitch Networks. Session Border Control in IMS - An Analysis of the Requirements for Session Border Control in IMS Networks, White Paper, 2011.
11. Fraunhofer FOKUS. Links for IMS Developers. http://www.openimscore.org/links
12. Richard Gayraud. Welcome to SIPp. http://sipp.sourceforge.net/
13. Lua Documentarion. http://www.lua.org/docs.html
14. Sebastien Blavier and Simon Tatham. ExtraPuTTY. http://www.extraputty.com/
15. T. Wilson, sniff2sipp, Digium, 2008

16. Miroslav Voznak, and Jan Rozhon: Approach to stress tests in SIP environment based on marginal analysis. Telecommunication Systems 52(3): 1583-1593, 2013.
17. Xiaoping Che and Stephane Maag: A Formal Passive Performance Testing Approach for Distributed Communication Systems. ENASE 2013: 74-84, 2013.
18. M. Zubair Rafique, M. Ali Akbar, and Muddassar Farooq. Evaluating DOS attacks against SIP-based VoIP systems. In Proceedings of the 28th IEEE conference on Global telecommunications (GLOBECOM'09), Mehmet Ulema (Ed.). IEEE Press, Piscataway, NJ, USA, 6130-6135, 2009.
19. Xiaoping Che and Stphane Maag: Passive Testing on Performance Requirements of Network Protocols. AINA Workshops 2013: 1439-1444, 2013.
20. Hemanth Srinivasan and Kamil Sarac. A SIP security testing framework. In Proceedings of the 6th IEEE Conference on Consumer Communications and Networking Conference (CCNC'09). IEEE Press, Piscataway, NJ, USA, 1056-1060, 2009.
21. Miroslav Voznak and Jan Rozhon. SIP registration stress test. In Proceedings of the 6th international conference on Communications and Information Technology, and Proceedings of the 3rd World conference on Education and Educational Technologies (WORLD-EDU'12/CIT'12), World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 101-105, 2012.
22. Miroslav Voznak and Jan Rozhon. Methodology for SIP infrastructure performance testing. W. Trans. on Comp. 9, 9 (September 2010), 1012-1021, 2010.
23. S. McGann and D. C. Sicker. An analysis of security threats and tools in SIP-Based VoIP Systems. In 2nd Workshop on Securing Voice over IP, June 2005.
24. M. Ranganathan, Olivier Deruelle, and Doug Montgomery. Testing SIP call flows using XML protocol templates. In Proceedings of the 15th IFIP international conference on Testing of communicating systems (TestCom'03), Springer-Verlag, Berlin, Heidelberg, 33-48, 2003.