A Distributed Architecture For Evaluating SIP Performance

Alain Vouffo-Feudjio and Ina Schieferdecker

Fraunhofer Institute FOKUS,
CC TIP
Kaiserin-Augusta-Allee 31, D-10589 Berlin, Germany,
phone: +49 30 3463-7000, fax: +49 30 3463-8000,
email: alain.vouffo@fokus.fraunhofer.de
http://www.fokus.fraunhofer.de/tip

Abstract: The Session Initiation Protocol (SIP) has been continuously gaining acceptance over the last years, as a valuable alternative to ITU-T's H323[1] in providing call signalling capabilities for audio, video and other multimedia communication over Internet Protocol (IP) based networks. With the number of SIP implementations growing at an extremely fast rate, the need for testing suites with regard to conformance, interoperability and QoS is becoming more and more crucial. While interoperability and conformance have been discussed and tested by several related work and test suites, the performance aspects of SIP implementations have (amazingly) been granted fairly low attention. We present in this analysis an architecture for evaluating the signalling performance of SIP implementations.
.

Keywords: SIP, Signalling, Performance, Testing, VoIP, TTCN

[1]H-323 is a standard specifying the protocols, procedures and components that provide multimedia communication services - real-time audio, video and data communication - over packet networks, including Internet Protocol (IP) -based networks

1 INTRODUCTION

Interoperability, i.e the ability of different vendors' implementations of a certain standard to interact properly with each other, is a key aspect in the computing and communication industry. Conformance Testing is generally viewed as a fundamental step towards ensuring interoperability. In fact if all systems implementing the standard are compliant to its specification, there are bigger chances for interoperability between those systems. That is probably the reason why most of the first efforts on SIP testing focused on that aspect. The SIP Interoperability Testing events (SIPIT)[2] and Testing Tech's TTSuite-SIP[3] are examples of such SIP conformance test activities.

On the other hand, the performance aspects of the SIP implementations, i.e after their conformancy has been tested successfully, seem to have been granted fairly low attention. In fact performance studies on Internet Telephony have mostly focused on voice quality, with call signalling performance being rarely discussed. Apart from Agilent Technology's Advisor tool for SIP described in [8], the only test suite to our knowledge dealing with SIP performance issues is SIPstone, currently being developed at Columbia University and described in [19]. While SIPstone is a benchmark attempting to measure the request handling capacity of a SIP proxy or redirect server or a cluster of such, Agilent's Advisor is rather a tool for analyzing network-traffic and providing support for SIP troubleshooting.

The subject of this work is to provide a general and extensible framework for testing SIP signalling performance. Instead of defining a composite benchmark to be computed from the collected performance data like SIPstone does with its SIPstone-A benchmark, our SIP Performance Framework provides the mechanisms for measuring and visualizing the signalling performance and leaves it to the test operator to evaluate it through comparison either with standard bodies' requirements [18] or with the results of other vendors' products. Such measurements could also be used for planning, provisioning and optimization of network capacity as well as for defining performance guarantees to customers through contractual agreements. This implies that the testing scenarios, metrics and parameters are close to those defined in SIPstone and similar work [21]. Furthermore, we take (as far as possible) into account the concepts of the Testing and Test Control Notation (TTCN-3 [10]) in order to enable the reuse of functional tests for SIP already defined within TTCN-3. This will address functional as well as performance tests based on the same technology TTCN-3[4].

The paper is structured as follows: Section 2 discusses the concepts of evaluating SIP signalling performance. Section 3 provides a description of our approach for a flexible and distributed test architecture for SIP performance. Section 4 describes the implementation of the concepts presented in section 3 and the results obtained for selected scenarios. Conclusions complete the paper.

2 CONCEPTS OF SIP SIGNALLING PERFORMANCE

The signalling performance of VoIP components always depends on the performance of the IP network they work on. Therefore, testing VoIP signalling performance should always involve assessing the performance of the lower layers components first. The metrics and requirements discussed in this chapter therefore do not focus on the IP-performance issues as such, given the fact that a high number of previous works have discussed that issue thoroughly. We assume that before the signalling performance of VoIP components is tested, the supporting IP-network's performance should have been optimized in a preliminary step.

Based on ISDN and SS7 standards published by ITU-T and on generic requirements proposed by Telcordia Technologies (formerly Bellcore) [5], the IETF defines a generic methodology as well as a set of requirements and metrics for evaluating VoIP signalling performance[18]. Therefore our approach consisted in mapping those requirements and metrics to the SIP-world and to provide a distributed testing architecture suitable for measuring them.

The remaining of this section provides a description of the metrics and parameters which need to be defined before getting into the details of our SIP performance testing approach.

2.1 Call establishment metrics

Call establishment is the key signalling objective as delays in this phase of a call are absolutely critical.

[2] www.columbia.edu/~hgs/sip/sipit
[3] www.testingtech.de/products/TTsuite-SIP/index.html
[4] An abstract test suite reflecting the concepts presented in this document has already been defined and a TTCN adapter is currently being developed to complete the TTCN-3 integration of the concepts.

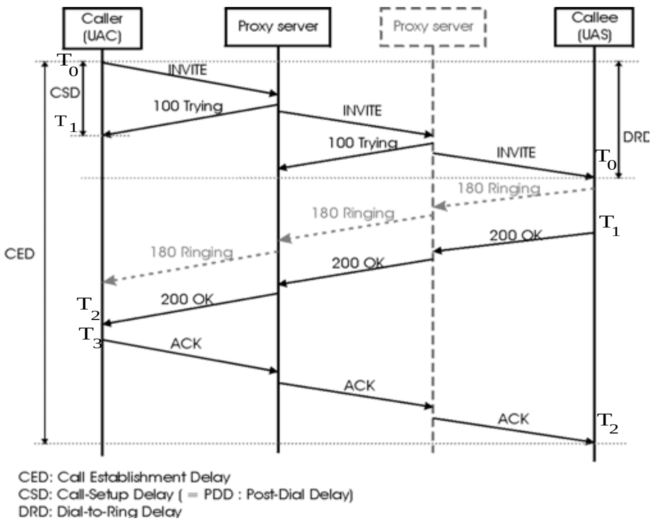

Figure 1: SIP call establishing procedure

The diagram on Figure 1 illustrates a typical SIP call setup procedure involving one or several SIP proxy servers and two user agents. In order to initiate a call, the user agent client (UAC) sends an INVITE request to the user agent server (UAS). This request is forwarded through the proxy server(s) until it reaches the destination UAS. The latter replies with a 200 OK final response to accept the incoming call. The call is finally established when the ACK request sent by the calling party for the 200 OK final response it previously received reaches the callee. Subsequently, a RTP or RTCP connection is created for the transmission of the voice or video data. Also, upon receiving the INVITE request, the UAS might first reply with a temporary response (e.g 180 Ringing) before sending the final response for that request. Using Basrur's [5] approach and based on ITU-T's recommendations several metrics can be derived from this diagram.

The call setup delay (CSD[5])

Eyers and Schulzrinne [11] defined the call setup delay in the traditional circuit-switched telephone network as the interval between entering the last dialled digit and receiving ringback. Even though this definition might lead to some confusion in the context of SIP (see below), we will adopt it for the sake of simplicity.

If we map this definition into the SIP context, the CSD could be defined as the interval between the time the caller sends an INVITE request and the time it receives the first temporary response (e.g a TRYING message) for that request. The confusion mentioned above stems from the fact that the temporary response used as the basis for calculating the CSD might originate from various sources. As illustrated on Figure 1 the temporary response might be sent by a proxy server before forwarding the INVITE request to the next proxy server in the path or to the callee's UAS, or it might be sent by the callee's UAS.

The call establishment delay (CED):

A very illustrative metric for signalling performance is the amount of time needed for a call to be established. Normally one would expect that the call setup delay would be the terminology used for this metric, but as stated above, it is generally used to represent a totally different value. For that reason, and since there seems to be no standard appellation for it, we will refer to this interval as the call establishment delay (CED).

In the SIP context, this would map to the interval between the caller sending an INVITE request and the callee receiving the ACK-request for a 200 OK response previously sent for that invitation.

Dial-to-ring delay (DRD[6]):

The Dial-to-ring delay is introduced by Eyers and Schulzrinne [11] as the amount of time expiring between a caller finishing to dial the callee's number and the moment the callee's phone would actually be ringing. For SIP it can be defined as the interval between the time the caller UAS sends an INVITE request and the time when that request actually reaches its destination, i.e the callee's UAS.

2.2 Call termination metrics

[5]The Call setup delay is also referred to as post-dialing delay [18] or post-selection delay [17]
[6]The Post-dialing delay is also referred to as "Call Answer Signal Delay" (CASD) in some literature.

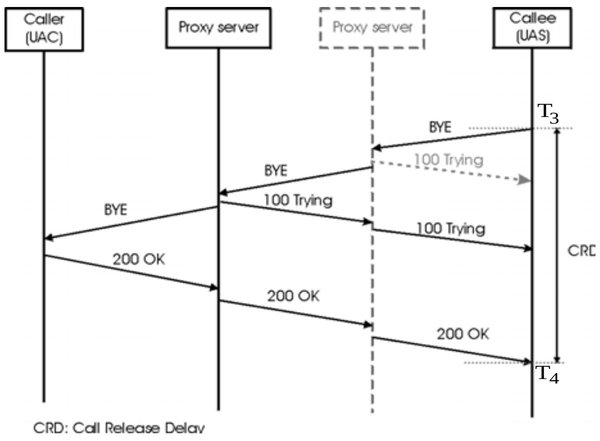

Figure 2: SIP call termination procedure

Figure 2 illustrates a SIP call termination procedure with the same components involved as in the call setup procedure discussed in Section 2.1. In order to terminate the call, one of the parties (in this case the UAS) sends a BYE request to the other party. The BYE request is processed by each proxy server it reaches and forwarded until it reaches its destination UA. Again the proxies receiving the BYE request might issue temporary responses to the sender to indicate the progress of the processing. Upon receiving the BYE request, the UAC replies with a 200 OK response to agree on the call being released.

Even though the delays in the call-termination phase are not as critical as the ones in the call setup phase, they cannot be considered as irrelevant at all. In fact they might radically affect the system's overall performance. E.g if the delays in releasing calls are too high, a SIP proxy server would have to keep greater amount of entries in its data base of ongoing calls, which might deteriorate its speed in handling incoming messages.

The call release delay (CRD)

The call release delay is proposed by ITU-T as the metric for signalling performance in the call termination phase. It is defined as the interval between the time one of the participants to the call hangs on the phone and the time the other participant hears the busy tone.

For SIP, the CRD can be defined as the lap of time between the moment a UA sends a BYE request and the moment it gets the final response for it.

2.1 Global metrics

Global metrics are the performance characteristics that cannot be directly associated to any of the call phases. They reflect the overall signalling performance in both the call setup phase and the call terminating phase. They are mostly statistics on performance-related failures in the SUT's (system under test) functional behavior.

The call success failure probability (i.e the probability that a call would not be successfully established or terminated) and further metrics such as those related to bulk signalling and termination are not considered here.

3 A distributed test architecture for SIP signalling performance

The requirements for performance tests are transparency, scalability, flexibility and repeatability. The Wisconsin Proxy Benchmark [1] is a good example of such an implementation, which is used for testing the performance of HTTP proxy servers. Due to the high number of conceptual and syntactical similarities between HTTP and SIP, we decided to use a similar approach for testing the signalling performance of SIP components. Figure 3 illustrates how this architecture can be used for testing SIP proxy servers.

Our testing architecture consists of one main test component (MTC) and several parallel test components (PTC). The MTC is the coordinating instance and the PTCs are directly interacting with the SUT. Each PTC is connected to the MTC at its coordination port and to the SUT at its SIP port. The coordination ports are used for communication purpose between the MTC and the PTCs. Using a set of predefined coordination messages, the MTC can send commands to PTCs and thus control their behavior. In the same manner, the MTC might receive a request from a PTC willing to retrieve some status information about the testcase as a whole or about another PTC. The coordination channel (i.e the communication channel between the MTC and a PTC over each coordination port) can be implemented as a TCP or UDP connection. In the later case, a time-out-resent mechanism has to be put in place to compensate packet loss.

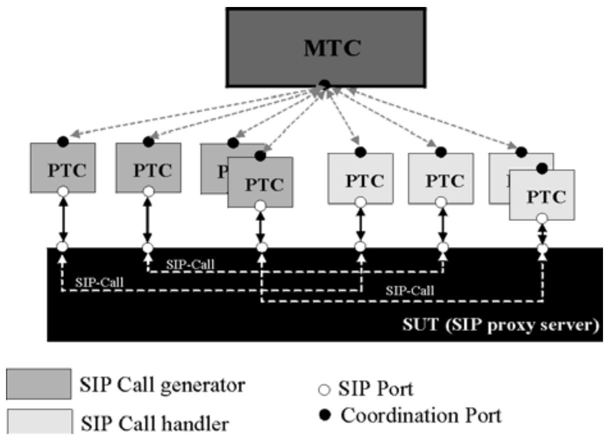

Figure 3: SIP performance testing architecture (SIP proxy is SUT)

For testing SIP proxies, each of the PTCs is assigned either the role of a caller or a callee. Following the instructions from the MTC (in form of coordination messages), the caller-PTCs generate SIP messages and send them to the callee-PTCs through the SIP proxy server.

3.1 Synchronization and Coordination of test components

Distributed testing architectures raise two issues that are absolutely crucial for proper results: synchronization and coordination.

Synchronization refers to the fact that for accurate time measurements, all the PTCs involved in a distributed performance testing architecture should have their clocks synchronized with each others. The most common way for achieving this is by the usage of a common clock or a similar mechanism. If this is not done, it would not be possible to correlate the event times recorded at the PTCs with each other to calculate the required performance metrics. On the other hand, coordination is the ability to control the behavior of the PTCs in the distributed testing architecture in order to trigger the appropriate actions and realize the intended testing scenario.

The synchronization of the PTCs could be implemented by placing all the PTCs reside on the same host and therefore use the same local clock or by setting a common external clock for all the machines where the PTCs will be running. However, by running all the PTCs on the same machine, performance bottleneck could occur with an increasing number of PTCs and leading to false measurements or to a crash of the test system. It is therefore recommended to use a common external clock as described previously, so that the PTCs could be deployed on several hosts. The Network Time Protocol (NTP), a standard-based (RFC 1305 [15]) method of maintaining accurate synchronization across multiple platforms, could be used for such purpose. It would enable us to synchronize the clocks of all the hosts participating in the test session with ms-accuracy. This can be regarded as acceptable with regard to the range of the time values we expect for our measurements, i.e above 100ms.

The MTC uses coordination messages to trigger some action or behavior at the PTC. The definition of what actions are to be undertaken at the PTC, when a certain type of message is received, is done at the PTC. For testing SIP proxies, we define six types of coordination messages:

SET messages: *SET* messages can be used to let the PTC perform some type of preamble in order to get in a certain state before starting its actual behavior. There are no options for SET messages. In case of testing SIP proxy servers, when receiving a *SET* message, a PTC will send the SUT a REGISTER-request, so that SIP calls addressed to it will be forwarded properly

GO messages: *GO* messages are used to tell the PTC to start its predefined behavior. If the PTC is supposed to play the caller-role (i.e is a call-generator), it will start making calls immediately. If it is supposed to be a callee, it will simply start its call-handler in order to reply to the incoming SIP calls. *GO* messages might contain options indicating some details on how the PTC should execute its behavior. E.g an option might be added to a *GO* message to define a timer-value, so that the PTC would execute its behavior for the specified time value and stop without any further notice. Also an option might be added to specify the number of calls to be made or the frequency at which calls should be made.

STOP messages: Upon receiving a STOP message a PTC interrupts any behavior it might be executing as soon as possible and terminates. Therefore the STOP message is generally sent by the MTC to stop the testing session.

REGISTER messages[7]: When a PTC is started, the first thing it has to do is sending a REGISTER message to the MTC. The *REGISTER* message contains information such as the the location of the host on which the PTC has been started, the port at which it will be receiving/sending SIP messages, the port at which the it will be expecting coordination messages etc.

Using this information, an entry is created for the registered PTC in a list maintained by the MTC. The index attributed to this PTC is returned as an option in the response to the REGISTER message.

PTCINFO message: In order to retrieve information about another PTC with whom it would like to exchange SIP messages, a PTC sends the MTC a PTCINFO message with the index of the partner PTC as option. If a PTC has already registered to the MTC and been attributed that index number, the information contained in the PTC's entry out of the MTC's list of PTCs is sent to the requiring PTC as option in the response.

Upon receiving a coordination message, a testing component replies with a coordination response. If the message was processed successfully an ***OK*** response is sent, possibly containing some options as discussed above. If not, a ***NOK*** response is sent. There are no options for NOK responses. They simply indicate that something went wrong and that the testing component's further behavior should take this into account.

3.2 Test Scenarios

Using the coordination message mechanism described above it is possible to elaborate test scenarios appropriate to evaluate the SIP signalling performance for various types of SIP components.

a) Proxy scenario

In the proxy scenario, the callee PTCs register to the SIP proxy server under test notifying it that further requests addressed to them should be forwarded to their contact-address, i.e a combination of their host-IP-address and their SIP port. In this case, all SIP messages are exchanged with the caller PTCs through the proxy server.

All metrics described in Section 2.1 can be measured in this scenario. The parameters in this scenario are the type of CPU of the SUT host machine, as well as the type of IP-connection between the PTCs and the SUT.

The population size is the number of PTCs participating in the test session. Due to the flexible test architecture introduced above, the performance measurements can be conducted under constant user population (i.e. the number of PTCs remain constant for the whole testing session) or under variable user population.

b) Redirect scenario

In the redirect scenario, the SUT is a SIP proxy server supporting redirection. After the begin of the tests, the callee PTCs register to the SUT notifying it to redirect further requests addressed to them to their contact address. The flow diagram on **Fehler! Verweisquelle konnte nicht gefunden werden.** illustrates the interaction of the caller PTCs with the SUT and the callee PTCs in this case, as well as the metrics that can be gathered during the test session.

The parameters for the redirect scenario are the same as for the proxy scenario described above.

Only the metrics for call establishment are meaningful in the case of redirect scenario, because the SUT is not involved in the call release process any more.

[7]REGISTER coordination messages should not be confused with SIP REGISTER requests sent to registrars.

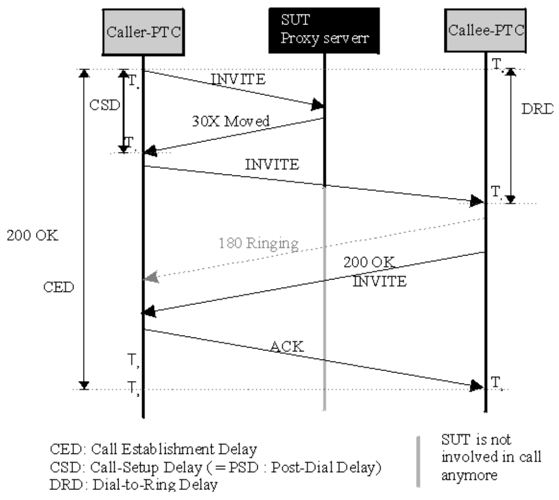

Figure 4: Redirect scenario

c) UAS scenario

In the UAS scenario, the SUT is a UAS. The testing architecture is the same as for the two previous cases, but all PTCs involved in the test suite play the caller role. The UAS scenario could be used by UA vendors to test their implementation’s robustness or simply whether it meets the minimal requirements regarding response delays, or by end-users to predict call setup delay for a VoIP connection.

The metrics in the UAS scenario are the same as the ones in the proxy scenario.

4 TEST IMPLEMENTATION AND MEASUREMENT RESULTS

The testing architecture described in this paper relies on off-line analysis rather than on-line analysis. Each of the PTCs involved in a testing session keeps a log-file containing pre-defined events, along with the time at which they occurred. At the end of the testing session, all logs are collected and combined to calculate the performance data. The log-files and the extracted performance data should be formatted properly to ease visualization of the results, possibly in form of curves representing the calculated metrics. This is then the basis for deciding, whether the SUT meets the signalling performance requirements or not.

Figure 5 presents an overview of the testing platform[8]. We implemented our approach for testing SIP proxy servers with the goal that, an extention or adaptation to test further types of SIP components (such as UAS, redirect servers or registrars) would imply adding new scenarios to the platform only.

[8]To keep Figure 5 simple, the MTC and its connections to the PTC are deliberately not displayed here

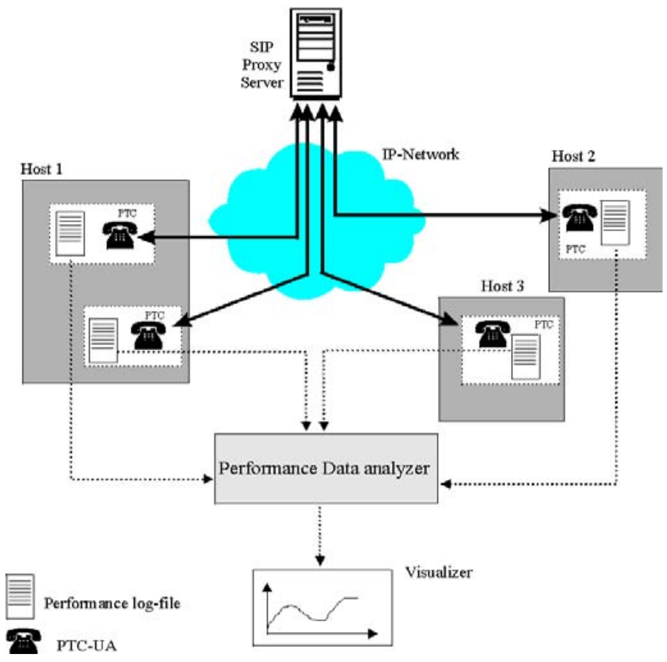

Figure 5: Platform overview

Using the approach described above, SIP signalling performance was tested on two example SUTs using the same hardware and based on the same IP-network connection.

The first SUT (SUT 1) was the example SIP proxy server shipped with the NIST-SIP software package, version 1.0 while the second one (SUT 2) was version 1.0 Columbia University's SIP proxy server sipd. Both servers were started successively on a Pentium III - PC, with 256 MB RAM running with the SUSE-Linux OS. The PTCs were left to run on another Pentium III machine running with the Redhat-Linux OS. Both machines were connected with a 10Mbit-ethernet link. Once again, we emphasize here on the fact that the results presented here should always be viewed in relationship to the testing environment and by no mean serve as absolute benchmarks for the SUTs' signalling performance.

4.1 Call Establishment

Figure 6 presents the CSD measurement results for the two example SUTs, with 100 calls for each call session under normal load. These results are equivalent to the ones the SUTs would obtain in SIPstone's proxy 200 tests. As visible on the graph, better results are obtained with SUT2, with the CSD being constantly below the recommended 200 ms limit. Although several values obtained with SUT1 exceed, its results can also be regarded as acceptable.

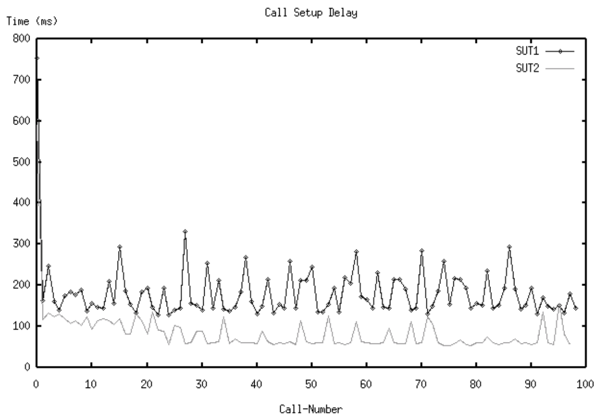

Figure 6: CSD-comparison for 2 example SUTs (normal load)

Figure 7 displays the results obtained for the CSD, this time under higher load. The number of participating PTCs has been doubled, which implies that each SUT must handle twice the number of calls of the previous case. As visible on that figure, SUT 1 delivers better results in that case with the CSD remaining at a constant level throughout the test session. This is a clear indication of stability. On the other hand, SUT 2 is clearly unstable under the given load and even crashes, so that the test session could not be resumed. This might indicate a performance bug, inappropriate configuration or memory management problems at the SUT

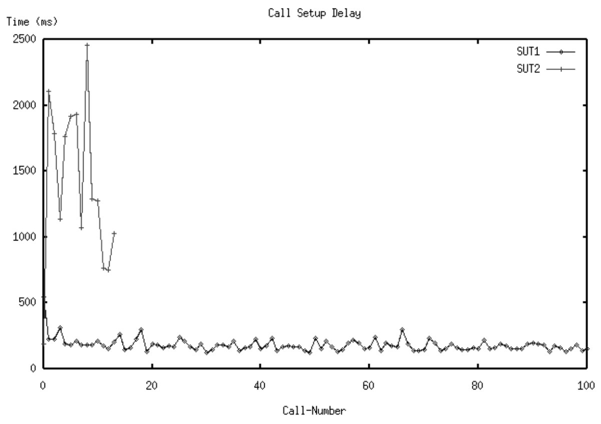

Figure 7: CSD-comparison for 2 example SUTs (High load)

Further call setup measurements have confirmed previous observations: the increasing delays due to higher load and a better performance of SUT2 with regard to response times and stability.

4.2 Call Release

Figure 8 presents the results obtained for the Call Release Delay with each of the two example SUTs. Although SUT 1 generates slightly better values, the difference between both SUTs' signalling performance at call release seems to be minimal. In fact the performance difference is not as big as for the call setup measurements

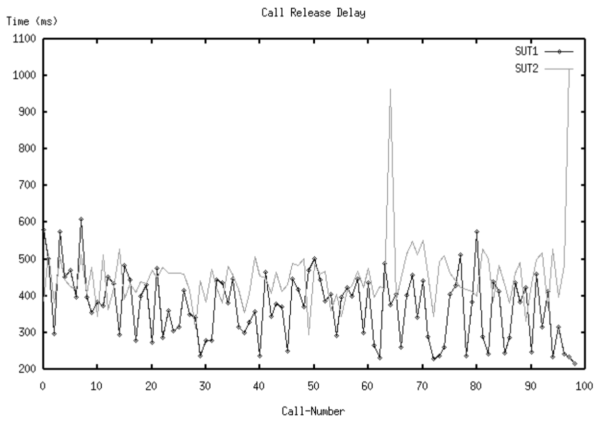

Figure 8: Comparison of CRD for 2 SUTs

5 CONCLUSIONS

This paper discussed the problem of testing signalling performance for VoIP systems, in particular for SIP implementations. A testing architecture appropriate for distributed and flexible test setup has been presented. The flexibility is in terms of SIP component to be tested, location of test components on nodes around the SIP implementation to be tested, number of test components and load to the tested SIP implementation. The test architecture has been implemented for SIP proxies and was successfully used to evaluate different SIP implementations. The results obtained for example SUTs and visualized through curve graphics show that the proposed approach is realistic and efficient.

In an extension of the work described in this paper, the usage of the standard Testing and Test Control Notation TTCN-3 to express that architecture and the test scenarios has been investigated. The application of TTCN-3 to performance testing enables the implementation-independent definition of well-defined, unambiguous performance tests, which allows to improve the transparency for the test process, to increase of the objectiveness of tests, and to enable comparability of test results. In addition, abstract performance test specifications can be adopted by the respective fora - in this case e.g. the SIP interoperability test event bodies.

The developed abstract test suite for SIP signalling performance in TTCN-3 is the basis for future work. One will be to examine the performance overhead generated by TTCN-3 by comparing the results obtained in this first phase with those of a derived ETS for example. Another will be to apply the test suite to further SIP components and test scenarios in order to investigate the applicability of the performance test concepts in a wider setting.

6 REFERENCES

[1] 3GPP, Technical Specification Group Core Network," TS 24.229 - IP Multimedia Call Control Protocol Based on SIP and SDP", Release 5, Jun. 2001

[2] J. Almeida, P. Cao, "Measuring Proxy Performance with the Wisconsin Proxy Benchmark", April 13, 1998

[3] American National Institute for Standards and Technologies (NIST), "NIST SIP/SDP Parser and Stack (v1.1) revI (02-20-02)", Online Documentation to the NIST SIP software package.

[4] J. Basrur, "Measurement Methods for VoIP Signalling Performance Estimation", ITU, T1A1-14 Standard project (Specification and Allocation of Internet Performance), October 2000.

[5] J. Basrur, "Objectives for VoIP Signalling", ITU,T1A1-14 Standard project (Specification and Allocation of Internet Performance), October 2000.

[6] U. Black: "Voice Over IP" - Prentice Hall Series in Advanced Communications Technologies - 1999

[7] S. Christensen, "Voice Over IP solutions", White paper, Juniper Networks, Professional services, June 2001

[8] T. Doumas (Agilent Technologies): "Next Generation Telephony: A Look At Session Initiation Protocol."

[9] ETSI STF 166/STF 176: "Proposal for Test Suite Structure (TSS) and Test Purposes (TP) for SIP" - Working Draft - 02.07.2001

[10] ETSI TTCN-3: Testing and Test Control Notation - Core Language. ETSI ES 201 873-1 V2.2.1, Aug. 2002.

[11] T. Eyers, H. Schulzrinne, "Predicting Internet Telephony Call Setup Delay"

[12] M. Handley, V. Jacobson, "SDP: Session Description Protocol", RFC2327, IETF, April 1998

[13] Handley, Schulzrinne, Schooler, Rosenberg: RFC2543 - "SIP Session Initiation Protocol" Internet Engineering Task Force Request For Comments - March. 1999

[14] Handley, Schulzrinne, Schooler, Rosenberg: RFC2543bis (-03) - "SIP Session Initiation Protocol" Internet Engineering Task Force INTERNET DRAFT - May.2001

[15] IETF, Network Working Group "Network Time Protocol (Version 3) Specification, Implementation and Analysis", March 1992

[16] ISO/IEC, "Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework", 15.07.1991

[17] ITU-T, "Network grade of service parameters and target values for circuit-switched services in the evolving isdn", Recommendation E.721, May 1999.

[18] H. Lin, T. Seth, A. Broscius, C. Huitema, K. Yang, "VoIP signalling performance requirements and expectations", IETF, Oct. 1999. Work in progress

[19] H. Schulzrinne, S. Narayanan, J. Lennox, M. Doyle, "SipStone - Benchmarking SIP Server Performance", August 9, 2001

[20] H. Schulzrinne, "The Session Initiation Protocol SIP", Slides on Internet telephony and multimedia, August 2001

[21] A. Vouffo-Feudjio: Signalling Performance in VoIP-Networks Testing Implementations of the Session Initiation Protocol SIP, MSc thesis, TU Berlin, FB Informatik, Febr. 2002

[22] X. Wang, H. Schulzrinne, D. Kandlur, D. Verma, "Measurement and Analysis of LDAP Performance"