Evaluating the Performance of an IMS/NGN Deployment

Dirk Thißen, Juan Miguel Espinosa Carlín, and René Herpertz

{thissen, espinosa, herpertz}@nets.rwth-aachen.de

Abstract:
The IP Multimedia Subsystem (IMS) is becoming the de facto overlay network for enabling the delivery of multimedia services in converged networks. Because it is envisioned that the IMS will experience a considerable growth in the years to come, it is important to verify that the involved resources conforming an IMS deployment deliver the desired performance when the system is operating under stress conditions. With the goal of gaining knowledge in the field of evaluation of such deployments, this paper introduces the IMS/NGN Performance Benchmark Specification developed by the ETSI, together with a detailed description of the results obtained when applied to an IMS testbed.

1 Introduction

For dealing with the challenge of offering better services that the ones offered by their IT counterparts, telecom operators developed the IP Multimedia Subsystem (IMS) as an overlay architecture to enable the delivery of rich multimedia services to end users, making use of the already available telecommunications infrastructure, and offering standardized signaling for provisioning services deployed in heterogeneous platforms.

With the aim of evaluating the performance of the core components of an IMS network, the ETSI developed the IMS/NGN Performance Benchmark Specification [Eur07], consisting of guidelines for applying a set of tests to determine how the system behaves when its load is increased. This benchmarking standard makes the benchmarking results comparable, which is an important step in taking decisions regarding the deployment of IMS systems.

With the goal of gaining knowledge in the filed of performance testing, this paper presents a performance evaluation based on the mentioned specification, using well-known open source software reference implementations and state-of-the-art analyzing tools for collecting the related measurements.

The rest of this paper is structured as follows. Section 2 presents the related work in the field, while Section 3 gives an overview of the IMS. Then, Section 4 describes the benchmark specification, while Section 5 presents the used enabling technologies. Next, Section 6 describes the deployment in which the testing was done, while Section 7 presents and discusses the obtained results. Finally, the conclusions and future work are briefly discussed in Section 8.

2 Related Work

In the field of benchmarking, considerable efforts have been done for defining accurate workload models. In [Bar04], the author explores different techniques for creating load models from the analysis of existing data, and presents a method for predicting actual performance when there is not enough information to model the production load. The authors from [MAM99] propose a methodology for characterizing and generating e-commerce workload models based on state-transition graphs, used to describe the behaviors of costumer groups that exhibit similar navigation patterns.

However, not much work has been on in the field of IMS benchmarking. The authors from [Din08] present an specification based on the Testing and Test Control Notation Version 3 (TTCN-3) from the ETSI. The paper gives the details about the implementation of the proposed approach, and a performance evaluation is presented in order to show the obtained results when the specification is used for benchmarking an IMS deployment. The benchmark execution lasts only 24 minutes, and the tested system is only evaluated with loads of 180 and 190 calls per second; moreover, no further information about the consumed computing resources is given.

3 IMS Overview

The IP Multimedia Subsystem (IMS) is a service delivery platform which establishes and controls Internet Protocol (IP) multimedia sessions between providers and customers. The development of IMS was mainly motivated by the idea of merging different network technologies to a layered all-IP network, with the possibility for providers to save costs by only managing a network of one kind instead of multiple kinds. The central components of the IMS are proxies and servers called Call Session Control Functions (CSCF's) and Home Subscriber Server (HSS), which form part of the IMS Layer depicted in Figure 1.

The main task of these components is to process the signaling needed for the provision of services or applications offered by the Service Layer. The Service Layer consists of servers, which can be of heterogeneous type, offering applications to the User Endpoints (UE) located in the Endpoints Layer. One of the main ideas of the IMS is the vision of having a network infrastructure ready to use any mobile device and at the same time to be able of running any application offered by the Service Layer. The Transport Layer takes care of transporting the signals through circuit-switched or packet-switched networks. Since circuit-switched networks are being continuously replaced by packed-switched ones, offering much higher data-rates, some requirements have to be guaranteed by the network infrastructure to achieve high quality telecommunication services.

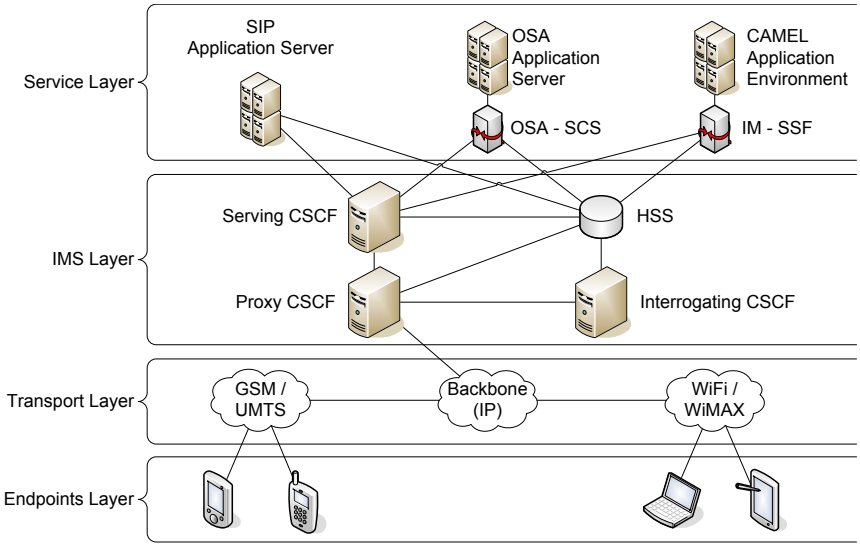


Figure 1: IMS Architecture

3.1 Quality of Service (QoS)

Real-time communication services need a guaranteed and mostly predictable amount of bandwidth and as less delay in packet transport as possible. In a conventional packet-switched network like the Internet, which provides a best-effort service without controlling bandwidth, delays, and packet routes through the network, there is no entity which ensures a certain level of QoS. The provision of voice communication services as well as other multimedia services through packet-switched networks like Voice-over-IP (VoIP) without QoS might result in an annoying experience. At one moment the communication between two persons could be perfect, but seconds later it could become utterly devastating. The IMS therefore takes care of synchronizing session establishment with QoS provision so that users have a predictable experience [CGM06].

3.2 Charging

Depending on the type of communication a large amount of data has to be transferred through the networks causing high costs for the operators of the networks. These costs have to be forwarded to the producing customer. Since in conventional packet-switched networks there is nearly no possibility to determine the contents of data packets, operators typically charge the customer using the amount of data being transferred through the network. The IMS provides information about the service the customer is requesting and enables operators to use this information to charge the customer depending on the multimedia services used, which might be a more beneficial charging scheme for the customer.

3.3 Integration of different Services

Network operators might not want to restrict themselves to one service developer offering only a low number of multimedia services. Instead, they might want to mix up different multimedia services provided by different services developers. Therefore a standardized platform is needed be able to provide and to orchestrate these services. The IMS defines the standard interfaces to be used by service developers [CGM06].

As the IMS is intended to be a main entity providing these requirements, telecommunication service providers have to implement and evaluate such systems. Because recurring implementation or even bug-fixing of such a system could become very costly, implementations have to be evaluated before deploying in order to gain sufficient operating experience. The ETSI has developed a specification about IMS/NGN Performance Benchmark, which was released in 2007, in order to define certain reasonable ground rules for defining an architecture and comparing it with others.

4 Performance Benchmark Specification

As already mentioned, the ETSI developed a Performance Benchmark specification called IMS/NGN Performance Benchmark [Eur07]. This specification consists of three parts; the first one providing a general introduction to the environment in which the Benchmark exists, the second one providing the subsystem configurations, use-cases and design objectives corresponding, and the third one documents the benchmarks tests through definitions of traffic sets and traffic-time profiles.

By definition, a benchmark measures the behavior of a system when an increasing number of users require to be served at nearly the same time. A benchmark test consists of a Test System (TS), which simulates a vast number of UEs conducting in a specific way, and a System Under Test (SUT), which reacts towards the requests performed by the users of the Test System.

Regarding the SUT, the specification only references the IMS architecture from a logical perspective, in order to avoid the mapping of functional elements to hardware of software components. Conversely, the components of a real IMS deployment do not map directly into the elements of the reference architecture, a fact that makes the comparison of similar products complicated.

However, this problem can be simplified by determining specific product classes that commonly reference the same subsets of elements in the architecture. For classes defined in this way, it is then possible to define a common set of benchmarks. These classes, called IMS subsystems, are expected to experience considerable growth in the future, as the IMS increases its market penetration.

To proceed to a benchmark test from the description of a subsystem, it is required that a complete description of all the aspects of the subsystem relevant to the test is present. This description enumerates the elements of the reference architecture and all the external

reference points. It is important to point out that a configuration requires the specification of the deployed hardware and software elements, due to the fact that the behavior of IMS systems is still being studied, and it is of interest to understand how CPU and network bandwidth utilization behave during operation.

In regard to the TS, the specification defines three main functions for it. First, it must be able to execute use-cases' scenarios according to the defined traffic profile. Next, it can optionally emulate the network characteristics on the different interfaces, including network bandwidth, latency, and error rate. Finally, for the case in which it is necessary to pass protocol information between an SUT interface and another and the Test System is different for the interfaces, a synchronization mechanism must be enabled.

The specification defines a scenario as a single interaction sequence between two users. A scenario attempt is defined as an instance of a scenario executed by the benchmark test. It is referred as a scenario instead of a call, because of the diversity of possible message types. A very important term used is Scenario Attempts Per Second (SAPS), which describes the average number of scenarios that are initiated by the Test System.

The specification also describes that a test procedure should first perform a preamble, during which any action to initialize the SUT should be done, such as registering an amount of users against the IMS (Pre-Registration Phase) and performing transactions on the subscriber base to randomize data (Stir-Phase).

Once finished, the benchmark test is ready to start its test (Run-Phase) stressing the SUT with predefined amount of SAPS and increasing it after a certain elapsed time, until the number of Inadequately Handled Scenarios (IHS) exceeds a threshold. The Test System decides whether a scenario is counted as successful or not. This is done when a scenario attempt fails or exceeds the threshold values defined for its use case; the whole benchmarking procedure is depicted in Fig. 2.

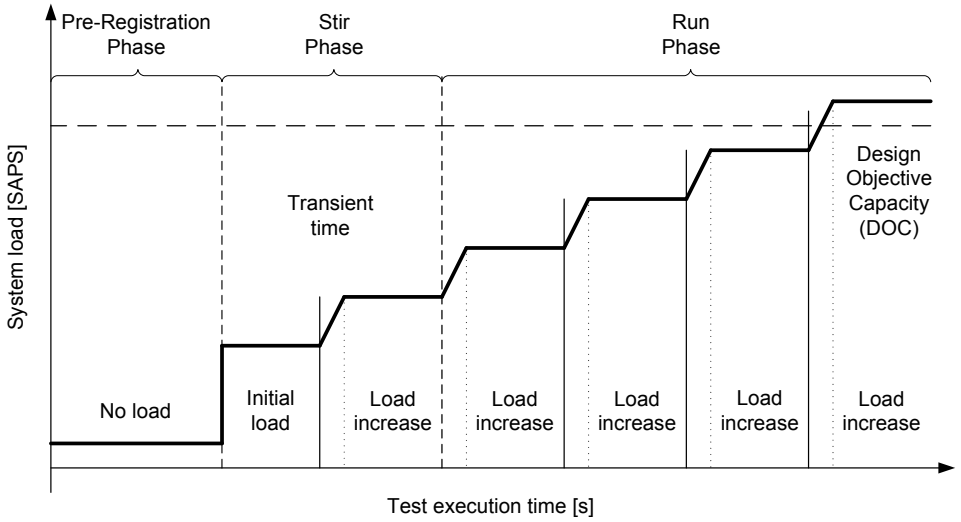


Figure 2: IMS Benchmarking Procedure

In order to consume resources of the SUT and so increase its workload, background traffic can be produced by an external system apart of the Test System as well as stressing the SUT by heavy local storage usage or other processing elements. When producing such a background load, the specification demands to document the type of system and software used to disturb the SUT. The source code of the software used and the configuration parameters of the Test System and the SUT should also be published in order to be able to replicate test circumstances.

5 Enabling Technologies

In order to be able to perform a benchmark test, implementation of an IMS as well as an implementation of the software and tools used by the Test System is needed. The following section will give a brief introduction to such public implementations we used for the evaluations wich will be presented in this paper later.

5.1 Open IMS Core Project

The Open Source IMS Core [Fra09] is an open source implementation of CSCFs and a lightweight HSS developed by the Fraunhofer Institute for Open Communication Systems (FOKUS). The implementation of the CSCFs is based on SIP Express Router (SER) [ipt06] written in C, while the HSS, named FOKUS Home Subscriber Server (FHoSS), is written in Java. Figure 3 depicts an overview of the components of the Open Source IMS Core. The Open Source IMS Core Project is not intended for commercial purposes; it should be understood as a reference implementation to the specifications related to IMS.

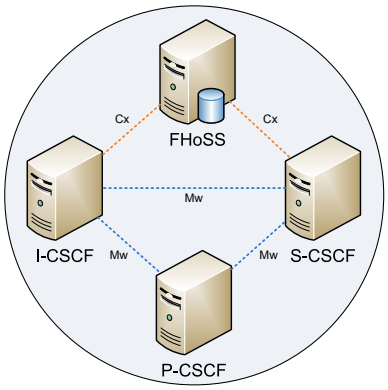


Figure 3: Open Source IMS Core [Fra09]

5.2 IMS Bench SIPp

The IMS Bench SIPp is a modified version of SIPp, a free open source traffic generator for the SIP protocol. The modifications were made in order to be able to handle complete scenarios using a large number of users, instead of only transmitting one single SIP transaction between two users.

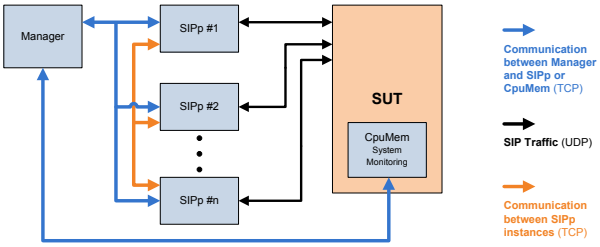


Figure 4: Test System High Level Overview [HP08]

In order to gather system information about the System Under Test (SUT) and the components of the Test System IMS Bench SIPp provides some tools to monitor and record these data as well as a report generation tool, which generates and calculates a detailed summary of the SUT's system state during the Benchmark run (i.e. memory and CPU data). The generated report is produced in accordance to the IMS/NGN Performance Benchmark specification, ETSI TS 186 008 [Eur07].

The Test System, as depicted in Figure 4, consists of one manager instance controlling the whole benchmark run, a fixed number of SIPp load generators, and a system monitoring tool for the SUT collecting information about CPU load and memory consumption. At the configuration of the benchmark test the manager assigns each SIPp instance a fixed number of users generating a predefined database containing user data as well as configuration scripts. It also generates a deployment script to deploy all needed files to the host machines of each SIPp instance. Each SIPp instance then generates SIP traffic towards the SUT in accordance with the statistical distribution of each scenario ordered by the manager instance and reports the total number of generated scenarios, the number of IHS and system information of the host machine. The manager instance decides if the SUT has reached its Design Objective Capacity (DOC) by means of the number of IHS divided by total handled scenarios.

IMS Bench SIPp is not only limited to IMS Benchmark testing; it is highly configurable and able to communicate with nearly any SIP application as long as the application operates correctly and, by means of call flow, predictably. However, there are also some limitations. Due to changes to the SIPp source code no other transport protocol than UDP is supported as well as IPv6 support and PCAP Play are not supported anymore. These limitations were accepted by the developers to speed up the implementation of IMS Bench and only have to be implemented sometime. Because the goal was to test the Open IMS Core Project and IMS Bench was specially designed for IMS Benchmark testing, this evaluation tool was used in the testbed.

6 Evaluated Deployment

Our testbed consisted of four hosts, one being in the role of the SUT (hostname `korell`), and the other three (hostnames `klejbors`, `solaria`, and `nexon`) building the Test System. The operating system of all four machines was the Linux distribution Debian. The hardware configuration of the machines used is presented in Table 1.

Hostname	CPU	RAM	Role
`korell`	2x2.8 GHz	2 GB	System Under Test (SUT)
`nexon`	2.2 GHz	1 GB	Test System (TS) Manager
`klejbors`	2.0 GHz	1.5 GB	TS SIPp
`solaria`	3.0 GHz	1.5 GB	TS SIPp

Table 1: Hardware configurations

The Open Source IMS Core, hosted by the SUT, was configured with the default parameters for listening SIP requests: TCP/IP port `4060` for the P-CSCF, TCP/IP port `5060` for the I-CSCF and TCP/IP port `6060` for the S-CSCF. Additionally, one of the host machines of the Test System was configured to act as a DNS server to properly resolve the default domain on which the testbed is running: `open-ims.test`. The FHoSS of the Open Source IMS Core was lined with a total of 100,000 subscribers in order to have a large number of users in the database available. For the purpose of reporting CPU usage and memory consumption of the SUT during the benchmark run, `cpum`, a tool part of IMS Bench and used to transmit this data to the Manager instance, was started before the test phase.

The Test System, consisting of one machine running the manager instance of IMS Bench, and four SIPp load generators, were placed in the same network as the SUT. In picture 5 a short overview is depicted.

The kernel of each host machine was recompiled with "Timer frequency" adjusted to 1,000 Hz in order to achieve timing measurements with millisecond precision. Also each host machine had to be synchronized, which was done by the use of the Precision Time Protocol deamon `ptpd`, which is available open source.

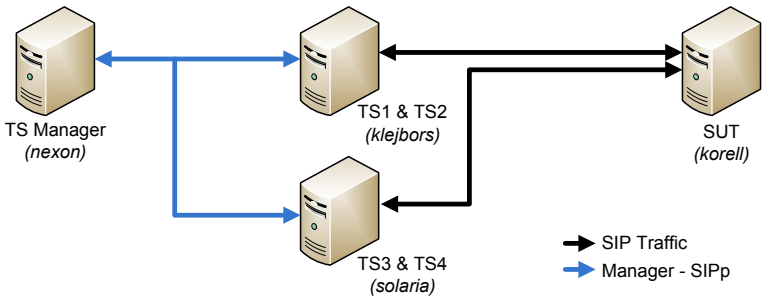


Figure 5: Testbed

Each load generator was configured to host two SIPp instances simultaneously using virtual IPs, which lead to a total of four SIPp load generators in the Test System. Each instance provided user data for 100,000 users to choose randomly during the benchmark test.

In order to have enough users registered at the time of the testing, IMS Bench was configured to register 40,000 users in Pre-Registration Phase with a rate of 15 calls per second (cps), a constant distribution of call attempts, and a maximum global IHS threshold of 10%.

The Stir Phase was configured to start at a call rate of 20 cps, a step increase amount of 20 cps, and a total of two steps. The duration of each step was set to 5 minutes, the distribution of call attempts was set to poisson, which is assumed to be the most realistic model of distribution in telecommunication, and the maximum global IHS threshold was set to 1%. The scenarios used were "calling" with a probability of 65%, which simulates the SIP messaging of the whole call setup between two subscribers, "messaging" with a probability of 30%, which simulates the SIP messaging between two subscribers based on SIP SIMPLE, and "registration" and "deregistration" with a probability of 2.5% each. The average call duration of the calling scenario was set to 2 minutes with a poisson distribution, and the average ring time was set to 5 seconds.

The Benchmark Run Phase was configured to start at a call rate of 60 cps using the same scenario mix as in the Stir Phase and maintaining the distribution of call attempts. Each step was configured to last 10 minutes before increasing the load by 20 cps. The maximum global IHS threshold was not set in order to be able to define thresholds for each single scenario, which were set to 0.1% each. The benchmark execution therefore stops as soon as the average percentage of IHS of any of the two deployed scenarios exceed the this threshold.

To allow each SIPp process to open a large number of sockets, some system limits had to be modified to avoid the "Too many open Files" error message. These are security settings of the underlying operating system, but for this case Benchmark execution without this modification is unfeasible.

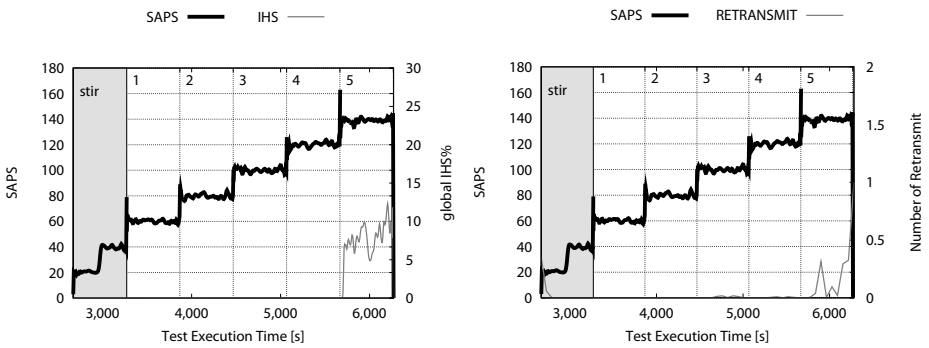
7 Performance Benchmark

The scope of this benchmark test was to determine the DOC of the Open Source IMS Core when subscribers make calls, send messages, and register and deregister in the IMS network; each scenario could be instantiated by any random user.

The total time for test execution was 1 hour and 44 minutes and the evaluated DOC was 120 cps. In the Pre-Registration Phase, a total of 40,000 users was registered by the Test System, which took 44 minutes and results in the preconfigured rate of 15 cps. The rate of the Pre-Registration Phase was chosen that low because of the experiences gained in earlier tests, which will be described later on. As the Pre-Registration Phase is not relevant for the scope of the benchmark test, it was left out from the results presented in the figures. The most left part of the graphs depicts the results measured in the Stir-Phase and it is labeled

at the top with the word “stir”. The dashed vertical lines divide each step, wick are labeled at the top with the number of step. As a reference, the bezier curve of the SAPS executed by the Test System is included in each of the following figures.

Having a closer look at the results, it can be observed the the SUT easily handled the tested scenario until the call rate passed the rate of 120 cps. In Figure 6(a) the bezier curve of IHS per use-case is depicted. The curve stays near to 0% for all steps until step 4.



(a) IHS (b) Retransmissions

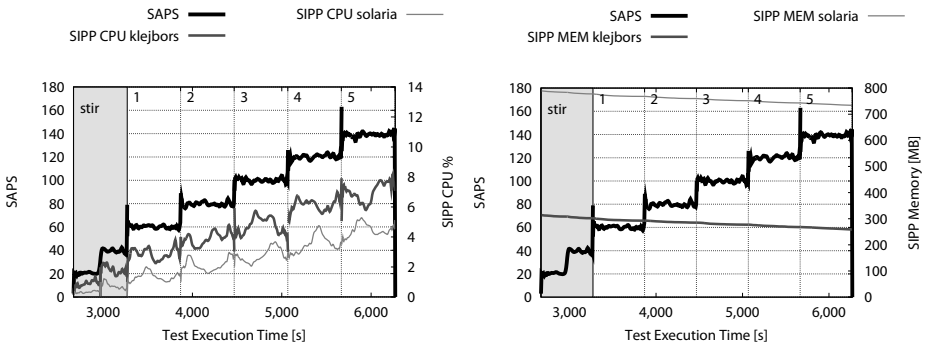
Figure 6: IHS and Retransmissions

As soon as the call rate increases to 140 cps, the number of IHS surges up to 12%, so the benchmark execution is stopped after completing step 5. Even the number of retransmissions, depicted in Figure 6(b), notably increases the beginning of step 5. In Table 2 can be seen that the mean percentage of IHS stays below the defined threshold of 0.1% for steps 3 and 4 for all use-cases; for step 5, the IHS reaches a maximum of 29.66%.

Step	cps	mean	min	max
3	100	0.00	0	0
4	120	0.00	0	0
5	140	7.52	0	29.66

Table 2: IHS per use-case [%]

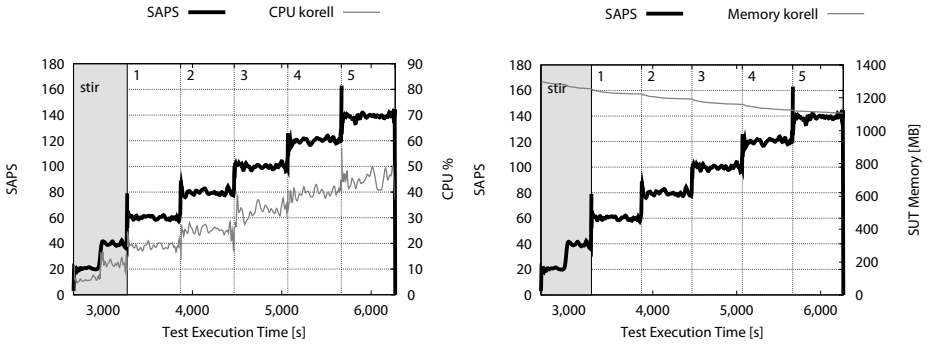
In order to further analyze the performance of both the Test System and the SUT, a closer look to the whole system was given. As already stated, the Test System consisted of four SIPp load generators in two physical hosts. In Figure 7(a) the CPU load of each machine hosting two SIPp load generators each is shown. The most important observation is that the CPU did not pass the 10% mark in any of the two hosts, although a clear increment in the CPU load was observed with every new step. Figure 7(a) clearly mirrors the used hardware as listed in Table 1, identifying `klejbors` as the powerful machine and `solaria` as the weak one. The other important parameter, the memory consumption of each machine, was also recorded by each SIPp instance and is depicted in Figure 7(b); as shown, none of the machines ran out of memory at anytime during the benchmark execution, a fact that rules out the possibility of the Test System causing the high IHS in step 5.



(a) Test System CPU (b) Test System Memory

Figure 7: Test System's CPU and Memory

Figure 8(a) depicts the load of the SUT's CPU at each step of the benchmark test. As shown, each time the call rate increases, the CPU usage of the SUT increases too. Moreover, the CPU load remains below 60% during the whole benchmarking time.



(a) SUT CPU (b) SUT Memory

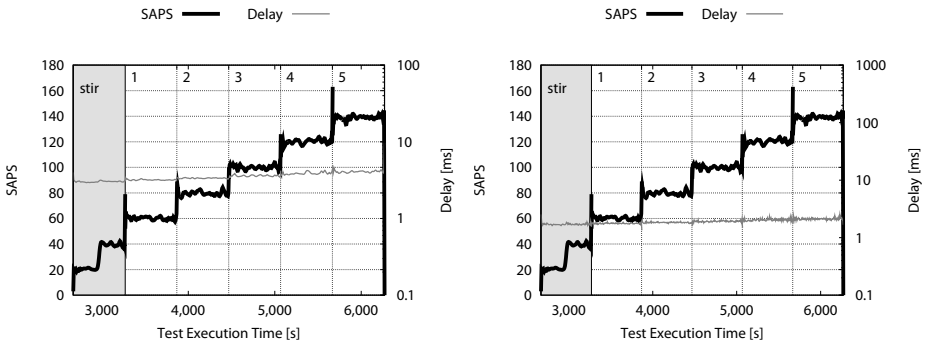
Figure 8: SUT's CPU and Memory

By having a closer look the the SUT's free memory, depicted in Figure 8(b), it can be concluded that the SUT never runs out of it, a fact that is confirmed by the values taken from the IMS Bench SIPp report and shown in Table 3: even with a load of 140 cps in step 4, the minimum available memory is 1,105.94 MB.

Step	cps	mean	min	max
3	100	1,168.65	1,159.52	1,192.25
4	120	1,136.00	1,124.44	1,159.52
5	140	1,113.63	1,105.94	1124.43

Table 3: SUT Available Memory [MB]

Finally, the benchmark specification also gives information about particular delays measured in the tested scenarios. Two examples of such cases are shown next. Figure 9(a) shows the session setup time in the "calling" scenario, defined as the delay between the caller sending the first `INVITE` request and the callee receiving the corresponding `ACK`, while Figure 9(b) shows the message transmission time in the "messaging" scenario, defined as the delay between a `MESSAGE` request and the correspondent `200 OK` response.



(a) Session Setup Time (b) Message Transmission Time

Figure 9: Test System's and SUT's CPUs

From these results, it can be seen that the experienced delays are proportionally affected by the number of SAPS. Table 4 shows the detail for both measurements.

Step	cps	Session Setup mean	Message Transmission mean
3	100	3.62	2.02
4	120	3.94	2.17
5	140	4.11	2.29

Table 4: Transmission Delays [ms]

Last but not least, it is important to point out that in order to verify that the previously presented results were consistent, the benchmark test was executed five times using exactly the same configuration. For all the independent test runs, the SUT behaved exactly the same way, quitting its operation after completing the execution of step 5. Moreover, further runs were made varying the IHS parameters per scenario, obtaining different results for the ones presented in this paper.

8 Conclusions

This paper introduced the ETSI IMS/NGN Performance Benchmark specification giving an overview of its goals, functions, definitions, and structuring of a performance test. Moreover, the paper presented the results that are obtained when such set of tests is applied to an IMS deployment based on well-known reference implementations.

As shown by the results, the Design Objective Capacity (DOC) of the used testbed was 120 cps, with an IHS of 0.1% for each one of the scenarios on the scenario mix, being the "registration" scenario the one responsible for the termination of the benchmark with a failure of 0.6% after step 5.

Regarding implementation, most of the found issues while configuring the testbed were related to the configuration of the Open IMS Core components, and it was found that the FHoSS, due to the programming language in which it is implemented, represented an important failure point for the whole system.

Future work in the area include the use of the performance benchmark for evaluating more complex scenarios including roaming and services based on extensions like SIMPLE. Moreover, IMS Bench SIPp is projected to be used while generating Presence and Instant Messaging traffic based on SIP with the aim of injecting it into an event-based service provisioning framework.

References

[Bar04] S. Barber. Creating Effective Load Models for Performance Testing with Incomplete Empirical Data. In *Proceedings of the Sixth IEEE International Workshop on Web Site Evolution*, pages 51–59, September 2004.

[CGM06] Gonzalo Camarillo and Miguel-Angel Garcia-Martin. *The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds*. John Wiley & Sons, Ltd, 2nd. edition, 2006.

[Din08] G. Din. An IMS Performance Benchmark Implementation based on the TTCN-3 Language. *International Journal on Software Tools for Technology Transfer (STTT)*, 10(4):359–370, August 2008.

[Eur07] European Telecommunications Standards Institute. Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); IMS/NGN Performance Benchmark. ETSI TS 186 008-1, October 2007.

[Fra09] Fraunhofer Institute for Open Communication Systems. OpenIMSCore.org — The Open IMS Core Project. `http://www.openimscore.org`, 2009. Last retrieved on the 15.01.2009.

[HP08] Hewlett-Packard. IMS Bench SIPp. `http://sipp.sourceforge.net/ims_bench/intro.html`, 2008. Last retrieved on the 09.03.2009.

[ipt06] iptel.org. SIP Express Router. `http://www.iptel.org/ser`, 2006. Last retrieved on the 15.11.2007.

[MAM99] Daniel A. Menascé, Virgilio A. F. Almeida, and Fonseca Marco A. Mendes. A methodology for Workload Characterization of E-commerce Sites. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 119–128, New York, NY, USA, 1999. ACM.