An Automated Passive Testing Approach for the IMS PoC Service

Felipe Lalanne*, Stephane Maag*, Edgardo Montes de Oca†,
Ana Cavalli*, Wissam Mallouli† and Arnaud Gonguet‡
* *Institut TELECOM & Management SudParis, CNRS UMR 5157*
9, rue Charles Fourier, F-91011 Evry Cedex, France
{*Felipe.Lalanne, Stephane.Maag, Ana.Cavalli*}*@it-sudparis.eu*
†*Montimage EURL, 39 rue Bobillot, F-75013 Paris, France*
{*Edgar.Montesdeoca, Wissam.Mallouli*}*@montimage.com*
‡*Alcatel-Lucent Bell-Labs, 54 rue de la Boetie, F-75008 Paris, France*
Arnaud.Gonguet@alcatel-lucent.fr

***Abstract*—Although the adoption of the IP Multimedia Subsystem (IMS) keeps growing, IMS applications are often integrated to the system without being formally tested. In this work, we are interested in the IMS Push over Cellular (PoC) service, an OMA standard. We propose a conformance passive testing approach to check that its implementation respects the main standard requirements. This approach is based on a set of formal invariants representing the most relevant expected properties to be tested. Two testing phases are applied: the verification of the invariants against the service specification and their testing on the PoC collected execution traces.**

***Keywords*-IMS, Testing, Formal model.**

I. Introduction

The IP Multimedia Subsystem (IMS) is a framework standardized to deliver IP multimedia services to mobile users over GPRS connectivity. Its main goal is to facilitate the access to voice or multimedia services in an access independent way for developing the fixed-mobile convergence. Besides, IETF standards such as the Session Initiation Protocol (SIP)[1] are also applied through the IMS for an Internet integration.

Since interoperated IMS platforms are becoming available, IMS applications (e.g. distributed billing, mash-up) are more and more numerous. Nevertheless, and because of its success, many protocol and service implementations were integrated without being formally tested. Therefore, failures (e.g. conformity and interoperability errors, security flaws, etc.) may be raised. The lack of testing in such areas may be explained by different reasons. One of them is the necessity for the industrials to quickly provide new services. To avoid that issue, we propose a passive testing approach to formally test an IMS service, the Push over Cellular (PoC)[1].

The rest of the paper is organized as follows. Section II describes the conformance testing approach as well as the concept of invariants used in our monitoring method. The Push over Cellular service and its formal SDL model are described in Section III. Then the industrial testbed and our experiments for testing the PoC are presented in Section IV. Section V provides related works and finally we conclude and present future work in Section VI.

[1]Research partially supported by the French Ministry of Research through the ExoTICus project, funded in the framework of the International ICT cluster System@tic program, and the SHIELDS project, funded by the European Community (FP7/2007-2013).

II. An Invariant-based Passive Testing Approach

A. The Concept of Invariant

The passive testing approach used in this work is based on invariant analysis, where the invariants are properties the Implementation Under Test (IUT) (in this work the PoC implementation) is expected to satisfy. An invariant is defined as follows:
Let $M = (S, I, O, s_{in}, f_{next}, f_{output})$ be an FSM (Finite State Machine) where S is a finite set of states, I a set of input actions, O a set of output actions, s_{in} an initial state, $f_{next} : S \times I \rightarrow S$ is the transition function and $f_{output} : S \times I \rightarrow O$ is the output function [2]. These two functions may be partially defined.

Formally, a sequence Inv is an invariant for M if the two following conditions hold:

1) Inv is defined according to the EBNF:

$$Inv ::= i/\bar{O} \mid *, Inv \mid i/o, Inv$$

where $i \in I \cup \{\theta\}$, $o \in O \cup \{\theta\}$ and $\bar{O} \subseteq O$.

2) Inv is verified on M.

Intuitively, a sequence such as $\{i_1/o_1, ..., i_{n-1}/o_{n-1}, i_n/o_n\}$ is an invariant for M if each time i_n/o_n is observed, then the trace $i_1/o_1, ..., i_{n-1}/o_{n-1}$ happens before. Invariants may be used to express properties where the occurrence of an event must be necessarily preceded by a sequence of events. In addition to sequences of input and output symbols, the wild-card characters 'θ' and '$*$' are allowed, where 'θ' represents any single symbol and '$*$' represents any sequence of symbols.

B. The Conformance Passive Testing approach

By conformance testing, we mean the process to test the correctness of an implementation through a set of invariants and observed traces (extracted from the running implementation). That process follows four main phases: (i) Properties definition. Standards or protocol experts provide the implementation properties to be tested. (ii) Properties as invariants. Properties are formulated as formal invariants. Besides, the properties may be formally verified on the formal specification guaranteeing that they are correct with respect to the requirements. (iii) Extraction of execution traces. A network sniffer is installed on one of the network entities. (iv) Invariants tested on the traces. The test of the expected properties (formulated as invariants) is performed and a verdict is provided (Pass, Fail or Inconclusive). An inconclusive verdict could be obtained if for instance the trace is too short in order to give a Pass or Fail verdict.

III. The Push Over Cellular Service and its Formal Model

A. The Push over Cellular Service

The IMS Push over Cellular (PoC) service, standardized by the OMA [3], is also known as Push-to-Talk, Push-to-View or Push-to-Share, depending on its main objective. It is an extension of the existing Push-to-Talk service over GPRS networks. It enables multiple IMS users to connect with each other in a single communication session, where any authorized user may talk simultaneously to every other participant. It is a walkie-talkie communication paradigm. This service is said to be half-duplex, meaning that at a given point a user can either talk or listen.

The PoC clients are integrated in the IMS clients and implement the functionalities required to get connected to the PoC server. Four consecutive steps are followed: (i) Initiate a new PoC session by sending a SIP INVITE message to the PoC server, which will be relayed to the invited participants. (ii) Handle incoming SIP INVITE messages to be involved in a PoC session created by another user. (iii) Quit a PoC session by sending a SIP BYE message to the PoC server. (iv) Handle PoC token requests by applying TBCP.

B. The PoC Formal Model

The IMS Push over Cellular has been specified in Specification and Description Language SDL, standardized by ITU-T [4]. The PoC functionality is divided into two parts or planes: the control plane and the user plane. The control plane deals with the communication between the different PoC entities, in particular, with the establishment of PoC sessions. The user plane deals with RTP media communication and floor control, referred to as Talk Burst Control, consequently, the floor control protocol used by the service being referred to as TBCP (Talk Burst Control Protocol), which is based on RTCP. In the service, the two planes interact in order to exchange session status information. In the formal specification developed, only the control plane is modeled, since only protocol part is relevant for the purpose of the paper. However, two TBCP messages are also considered for the model, *Talk Burst Granted* and *Talk Burst Taken*, used in order to signal the correct establishment of the session, and the floor to speak of the initiating client. In our model, all the parts in the communication process are considered, this means the PoC Client, IMS and PoC Server. The Figure 1 shows the overall architecture.

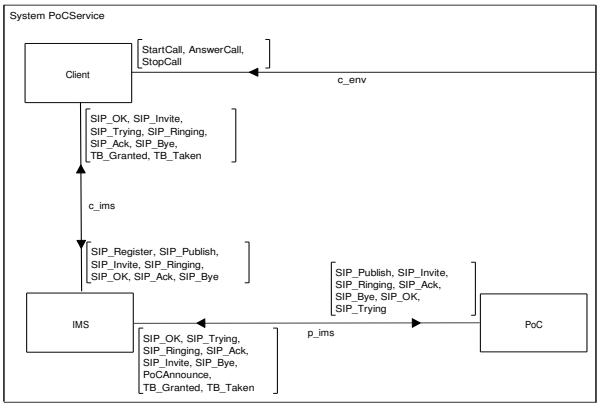

Figure 1. Overall architecture of the PoC formal specification

IV. Experimentation and Results

A. The Experimented Industrial IMS Architecture

Our approach has been applied on a real industrial IMS architecture designed to support 10,000 users, developed and provided by Alcatel-Lucent France. The IMS services run on an A5400 IMS Application Server (AS) processing the XDM server (the OMA 1.1 network address book manager), the Presence Server, the voice communication service , the voice mail server and the PoC server implementing the OMA 1.1 PoC service.

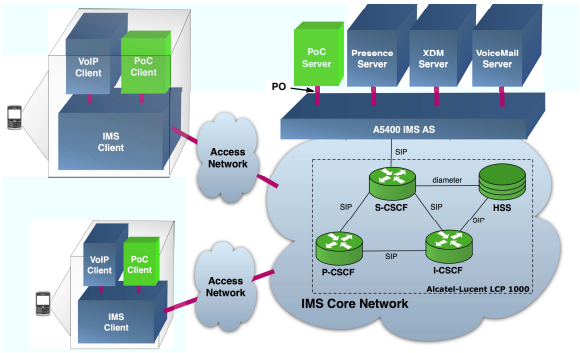

Figure 2. The Alcatel-Lucent IMS Architecture

Functionally, the PoC server is divided in two parts: the controlling PoC function managing the number of communication channels dedicated to the necessary streams (TBCP,

SIP) that has to be equal to the number of participants to the PoC session, and the participating PoC function which provides PoC sessions handling by relaying the TBCP messages between the controlling PoC function and the different PoC clients. The Figure 2 depicts the industrial IMS architecture. Furthermore, the PO illustrates the interface where all the messages used in the PoC service can be captured.

B. Invariants

From the OMA PoC requirements, six properties to be tested on the IUT have been provided. Each one of these properties describes a step in the session establishment for a PoC Ad-hoc session and allows ensuring that the PoC server follows the sequence of messages defined by the protocol [5]. According to the formal definition given in Section II-A, we describe these invariants in the following, as well as their verification process.

1) The Invariants Design:

Invariant 1:

- $INVITE(CSeq = c0, From = u0, To = Conference, Invitees = \{u1\})/\theta, *, \theta/INVITE(CSeq = c1, From = u0, To = u1, Invitees = \phi)$

This invariant illustrates the case where a particular user initiates an Ad-Hoc call. Since this session type corresponds to a quick call to the users selected from a list, the list of recipients or invitees is included in the invariant. For each of the users in the list, the PoC server initiates a new *INVITE* call, using the initiating user's URI in the *From* header. The invariant means that if an *INVITE* is observed being sent by the PoC server, then the PoC server should have received an invite to the conference URI initiated by the user $u0$.

Invariants 2, 3:

- $\theta/INVITE(CSeq = c1, From = u0, To = u1), *, Ringing(CSeq = c1, From = u0, To = u1)/\theta$
- $INVITE(CSeq = c0, From = u0, To = Conference)/\theta, *, \theta/Ringing(CSeq = c0, From = u0, To = Conference)$

These both invariants illustrate one of the steps in the session initialization sequence. After a SIP peer receives an *INVITE* request, it replies with a *Ringing* response to indicate that the message was received and that it is waiting for the user to accept the call. In the case of the Ad-hoc session, since at least two *INVITE* messages are required, then at least two *Ringing* messages are to be expected: one from the end client, indicating to the server that the message was received, and one from the PoC Server to the originating client indicating that at least one of the recipients has received the invitation. Both invariants indicate that if a *Ringing* message is received/sent, then a previous *INVITE* message must have been sent/received by the PoC server.

Invariants 4, 5:

- $INVITE(CSeq = c0, From = u0, To = u1)/\theta, *, \theta/OK(CSeq = c0, From = u0, To = u1)$
- $\theta/INVITE(CSeq = c0, From = u0, To = u1), *, OK(CSeq = c0, From = u0, To = u1)/\theta$

These invariants illustrate respectively the cases when a PoC client informs the PoC server that the call has been accepted by the user and that he is waiting to listen, and when the PoC server notifies the originating client that at least one of the recipients has confirmed. For both cases this is done via an *OK* response to the original *INVITE*. Both invariants indicate that if an *OK* message is sent/received, then an *INVITE* request must have been received/sent previously with the same parameters.

Invariant 6:

- $\theta/OK(CSeq = c0, From = u0, To = Conference), *, ACK(CSeq = c2, From = u0, To = Conference)/\theta$

This invariant illustrates the last step of the session establishment, after the PoC server has notified the originating PoC client that a user has accepted the call via an *OK* response, the PoC client must send an acknowledgment message in order to indicate that he is waiting to talk. The invariant means that if an *ACK* request is observed being received by the PoC server, then the PoC server should have sent an *OK* response previously.

2) The Invariants Verification:

A- The GOAL observers: The invariants may be verified against the formal SDL specification guaranteeing then that they are well designed and conform to the functional properties of the tested protocol.

To do it, we specified our invariants using the GOAL language (Geode Observation Automata Language)[6]. For each invariant, we create an observer designing the states, its signals leading to a triggered transition or a task sequence putting the Extended FSM (EFSM) in a new state. Afterwards, the GOAL invariants are verified on the PoC model by applying the ObjectGeode (OG) tool [7]. This tool provides a partial reachability graph of the EFSM model (i.e. an FSM) to automatically verify the properties on the model focused on the PoCCore process. The six above mentioned invariants have been verified by OG on the PoC formal model running three distinct clients.

B- Verification results: We used a computer processing an Intel Core 2 Duo with 4Mo L2 cache. Nevertheless, when checking the invariants by applying the observer described above, we had an extremely large number of explored states (16.777.214) and transitions (63.212.562) before covering 80% of the model. Despite that we generated a huge exploration file (more than 12GB) in an average

Inv	#States	#Transitions	Time	Coverage	Results
Inv1	6.586.011	34.452.783	0:37:31	100%	Verified
Inv2	12.358.441	57.952.942	1:18:43	100%	Verified
Inv3	9.713.917	44.618.348	0:57:09	100%	Verified
Inv4	8.932.488	39.025.924	0:47:10	100%	Verified
Inv5	10.472.401	48.043.571	1:09:39	100 %	Verified
Inv6	16.777.214	63.212.562	1:37:18	75.03%	Verified

Table I
VERIFICATION RESULTS

of 2 hours, we unfortunately met the state space explosion problem. We therefore checked the invariants one by one on the specification. The main results are presented in the Table I. No livelocks nor deadlocks have been detected.

C. Passive Monitoring and the TestINV tool

The traces of the IMS PoC service have been provided by Alcatel-Lucent France. Wireshark[2] was used to save the traces in XML format. Different traces have been extracted. And in order to optimize our testing process, a reset has been applied on the IMS core network and on the AS to allow the traces to contain the initial state. However, this *homing* state phase is not mandatory. Our approach makes it possible to inspect the invariants at any point of the implementation's execution. Nevertheless, as our invariants mainly tackle the initiation phase of the PoC sessions, we noticed that such kind of *network reset* accelerates the testing verdicts provided by the TestINV tool.

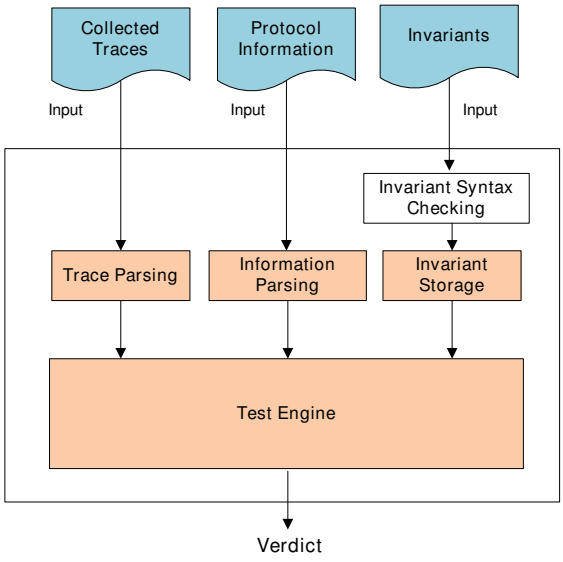

Figure 3. The TestInv Tool

The TestInv tool [3] processes automated analysis of captured traces to determine if the tested invariants are correct or not. The tool takes as input: needed protocol information, the traces and the invariants both defined in XML format. A high level description of the tool is illustrated by the Figure 3.

In order to use the tool, first the invariants and the protocol information need to be defined. This is performed by an expert of the protocol and needs to be done only once for each different protocol. The protocol information defines the data of interest that will be extracted from the traces and corresponds to the packet field names specific to the protocol that is being observed. This is needed to improve the performance and limit the amount of memory needed when analyzing the captured traces.

The algorithm used by the tool analyzes the traces in, at most, time complexity of $O(N^2)$ or to be more precise: the number of packets that need to be analyzed is $N^2 \times I$ where N = number of packets in the trace and I = number of invariants. This can be reduced to $O(N)$ if we store information of each condition for each packet in a hash table. Figure 4 shows the processing time as a function of the length of the trace.

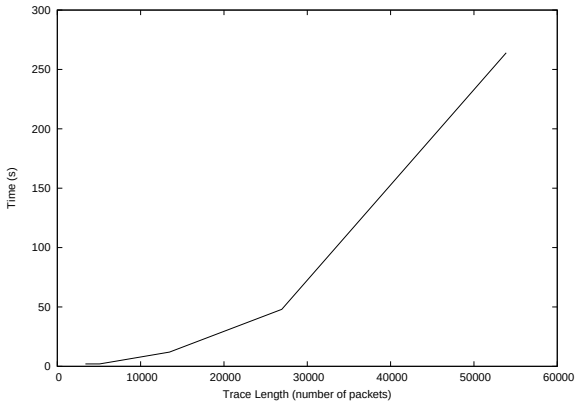

Figure 4. Processing time for the TestInv tool.

D. Results and Analysis

1) Experimental results: The invariants were defined using the TestInv XML format, specifying the types of packets to observe and the events to evaluate. Applying the tool to the traces was rapid (less than 1 sec.) since the traces did not need to be very long for these kinds of invariants.

The verdicts obtained were all PASSED except for some instances of the 4th and 5th invariants. However, we noticed that these FAILS were *false positives*. But, for each one of the invariants, at least one occurrence of the property was found in the traces, illustrating the correctness of the trace.

2) Analysis: The results obtained with the 4th and 5th invariants shows the difficulty, in some cases, of applying passive testing techniques to an application such as the PoC. Because of the design of the IMS architecture, applications can share information to avoid duplicating functionality. The PoC Server, as well as other applications, makes use of both

[2]http://www.wireshark.org/
[3]developed by Montimage

of these capabilities and, thus, communication packets from both of these applications will invariably appear in the trace. This, in addition to the User Plane information, makes it particularly difficult to distinguish the relevant information in the trace.

The invariants 4 and 5 are correct, because an *OK* message indicates the acceptance of the terms of the session establishment initiated by the *INVITE* request for the case scenario considered. Nevertheless, from a global point of view, the *OK* response also marks the acceptance of any request, for example, *SUBSCRIBE* and *NOTIFY*, used by the presence protocol. This way, when an *OK* message is found in the trace, it will look for an *INVITE* message whether it is actually a reply to this type of request or not.

V. Related Works

Although no work exists to formally test IMS applications passively, we may cite interesting research work on passive testing approaches. In the particular context of the EFSM semantic model, some important works can be mentioned, such as [8] and [9] where algorithms based on Event-driven EFSM are proposed. The expected properties are specified in a symbolic logic expression in order to be able to define in detail a set of valid variable values. Some works have been done in modeling SIP in SDL[10]. In [11], starting from that model, the authors show how to detect feature interactions by determining if a particular MSC (Message Sequence Chart) trace can occur in the provided model.

VI. Conclusions & Perspectives

This paper introduces a novel approach for testing IMS services, focusing in particular in the case of the Push-to-Talk Over Cellular service. This was motivated by the fact that it is not always possible to apply active testing techniques to protocol testing, especially when the platforms are already deployed or are closed implementations. In this paper we propose and evaluate an extensible and more flexible approach where properties can be added depending on the set of features of the tested service.

As future work, we plan to explore and evaluate new types of invariants in this framework. In particular we will tackle the issue raised by our Invariants 4 and 5 where the communication with other IMS entities produces the *false positive* results, although not an issue in the present work, this could be necessary to perform automated trace checking. We also intend to extend the developed model in order be able to test other kind of sessions, the PoC client's conformance and its interoperability with the PoC server implementation.

We also plan to adapt the tool to be able to analyze packets on-line. This will allow to use it to detect that the protocol exchanges occur as expected.

References

[1] M. Handley, H. Shulzrinne, E. Schooler, and J. Rosenberg, *RFC 2543 - Session Initiation Protocol (SIP)*, IETF, March 1999.

[2] D. Lee and M. Yannakakis, "Principles and Methods of Testing Finite State Machines - a Survey," in *The Proceedings of IEEE*, vol. 84, August 1996, pp. 1090–1123.

[3] *Push to talk over Cellular V2.0*, Open Mobile Alliance, oct 2007.

[4] ITU-T, "Recommandation Z.100: CCITT Specification and Description Language (SDL)," ITU-T, Tech. Rep., 1999.

[5] Open Mobile Alliance, "OMA PoC Control Plane. Approved Version 1.0," Jun. 2006.

[6] B. Algayres, Y. Lejeune, and F. Hugonnet, "GOAL : Observing SDL behaviors with GEODE," in *Proc. of SDL'95*, 1995, pp. 223–230.

[7] Verilog, *ObjectGEODE 4.0 - User Manual*, IBM, 2000.

[8] D. Lee, D. Chen, R. Hao, R. Miller, J. Wu, and X. Yin, "A formal approach for passive testing of protocol data portions," in *Proc. of 10th IEEE International Conference on Network Protocols*, 2002, pp. 122–131.

[9] D. Lee, D. Chen, R. Hao, R. E. Miller, J. Wu, and X. Yin, "Network protocol system monitoring: a formal approach with passive testing," *IEEE/ACM Transactions on Networking*, vol. 14, no. 2, pp. 424–437, 2006.

[10] K. Y. Chan and G. von Bochmann, "Modeling IETF Session Initiation Protocol and its services in SDL," in *SDL Forum*, ser. Lecture Notes in Computer Science, R. Reed and J. Reed, Eds., vol. 2708. Springer, 2003, pp. 352–373.

[11] K. Y. Chan and G. V. Bochmann, "Methods for designing SIP services in SDL with fewer feature interactions," in *Proc. 7th. Feature Interactions in Telecommunications and Software Systems*. IOS Press, 2003, pp. 59–76.