Interoperability testing of presence service on IMS platform

Mazen EL MAARABANI, Asma ADALA, Iksoon HWANG, Ana CAVALLI
Software-Networks Department, TELECOM & Management SudParis
9, rue Charles Fourier, 91011, Evry Cedex, France
Email: {mazen.el_maarabani, Iksoon.Hwang, Ana.Cavalli}@it-sudparis.eu

***Abstract*—In this paper, we perform experiments on interoperability testing of presence service on IMS platform. Open source implementations such as an IMS platform, a presence server, an XCAP server, and IMS clients are deployed to establish testbed for interoperability testing of presence service. Test cases for SIMPLE presence protocol developed by OMA are applied to our testbed and test results are analyzed in detail. In our experiments, we covered the basic functionalities of the presence service and the interaction between a client and an XCAP server for intra-domain case. And then we explained the testbeds for inter-domain cases and discussed briefly the differences with intra-domain case. As a result, we found that a number of functionalities could not be tested because they are not supported by IMS clients. Also we found a number of problems in presence server and IMS clients, which results in a number of fail verdicts for some test cases.**

I. Introduction

Interoperability is the ability of two or more networks, systems, devices, applications or components to exchange information between them and use this information [1]. The purpose of interoperability testing is to give higher confidence on interworking between at least two communicating systems as required by the standards. Although each implementation passes conformance testing, which is to verify that implementations conforms to specifications, we cannot guarantee that two implementations can interwork without any problem because they may have different implementation options, some part of specifications can be interpreted in a different way, they may be based on different versions of specifications, etc.

Interoperability testing on IMS testbeds consists of two parts: testing of interconnectivity between components and testing of service interoperability. In this paper, we perform experiments on interoperability testing of IMS services, in particular the presence service [2]. The choice of presence service has been motivated by different reasons: the presence service can be the basis of new multimedia services and communications; it allows inferring the context, availability and willingness of a user to accept or participate to a particular type of communication or activity. The presence service may also enable the creation of services in which abstract entities are providing information to mobile terminals. Abstract services can be cinema ticket information, the score of a football match, motorway traffic status, etc.

Due to the importance of the presence service, a number of industrial researches have been done for interoperability of the presence service. OMA (Open Mobile Alliance) released the test cases [3][4][5] and the test reports [6] for the interoperability of presence service with terminal equipments from different vendors. Nokia, Motorola, and Ericsson launched the Wireless Village initiative, which was consolidated into the OMA later, to define a set of universal specifications for mobile Instant Messaging and Presence Services (IMPS) and presence services for wireless networks [7].

In this paper, we deployed open source implementations such as Open source IMS platform from Fraunhofer FOKUS (Fraunhofer Institute for Open Communication Systems) [8], open source presence server based on OpenSER [9], open source XCAP (XML Configuration Access Protocol) server [10], and open source IMS clients [11][12] in order to establish testbed for interoperability testing of presence service. Test cases for SIMPLE (SIP Instant Messaging and Presence Leveraging Extensions) presence protocol [13] developed by OMA were applied to our testbed and test results were analyzed. Due to the non-supported functionalities by clients such as handling of presence composition rules, subscription/notification filtering, and partial presence information publishing, it was not possible to test all test cases defined by OMA. Also we had fail verdicts for some test cases because of a number of problems in presence servers and IMS clients.

In our experiments, it is assumed that all communicating components are in the same home network, which we will call intra-domain case. For the case when some components are located in different networks, e.g. presentity is in visited network, which will be called inter-domain case, we explain how to establish testbeds for such cases and discuss briefly the differences with intra-domain case. In summary, the main contributions of this paper are as follows:

- A complete testbed for presence service is established using open source implementations.
- Interoperability testing is carried out based on test cases developed by OMA and then results are analyzed in detail.
- Testbeds for the inter-domain cases are proposed for future work and the differences with intra-domain case are discussed briefly.

The paper is organized as follows. In section II, we explain interoperability testing process defined by ETSI and related works on interoperability test case generation. Section III briefly introduces the presence service. Test architecture and

deployment of testbed is illustrated in section IV. In section V, we explain the test cases that we applied and discuss the test results. Finally, section VI concludes the paper.

II. Interoperability testing

A. Interoperability testing process

In this section, we explain interoperability testing process defined by ETSI [1]. In the concept of interoperability testing from ETSI, it is assumed that we have Equipment Under Test (EUT) and Qualified Equipment (QE) where EUT and QE should come from different suppliers. Interoperability tests are then performed with normal user control and observations, i.e. there is no specialized interfaces for testing purposes and testing is based on functionalities that a user experiences. One of the important issues is to develop interoperability test cases. As a first step, it is necessary to identify interoperable functions and abstract test architectures. Once interoperable functions and abstract test architectures are identified, we can develop test purposes and define test cases. The steps for developing interoperability test cases are shown as follows:

1) Specify abstract architecture: This step defines abstract test architecture where EUT, QE(s), communication paths between EUT and QE(s), and, if necessary, the expected protocols to be used for communication paths should be clearly identified;

2) Prepare draft interoperable features statements: This step identifies whether each function in the standard is mandatory, optional or conditional by other functions;

3) Specify test suite structure: This step divides the test suite into test groups based on some logical criteria and defines test coverage within each test group;

4) Write test purposes: In this step, a full description of the objective of each test case is specified in its test purpose;

5) Write test cases: For each test purpose, detailed test steps that must be followed in order to achieve the test purpose are described. Test cases can be written in either natural languages (e.g. English) or test specification languages (e.g. TTCN-3 (Testing and Test Control Notation version 3)) or programming languages (e.g. C++) or scripting languages (e.g. Perl). If test cases are written natural language, they should be specified in a clear and unambiguous way.

Once we have interoperability test cases, we can perform testing process including test planning, test configuration, execution of tests, and producing test reports.

B. Interoperability test case generation

In this section, we explain related works on test case generation for interoperability testing. OMA is the leading industry forum for developing market driven, interoperable mobile service enablers. OMA has developed a number of mobile service specifications including test specifications in order to support the creation of interoperable end-to-end mobile services. Based on their test specifications, testing events, called OMA TestFests, are held regularly for interoperability testing of services such as VoIP, push to talk, presence and instant messages [14].

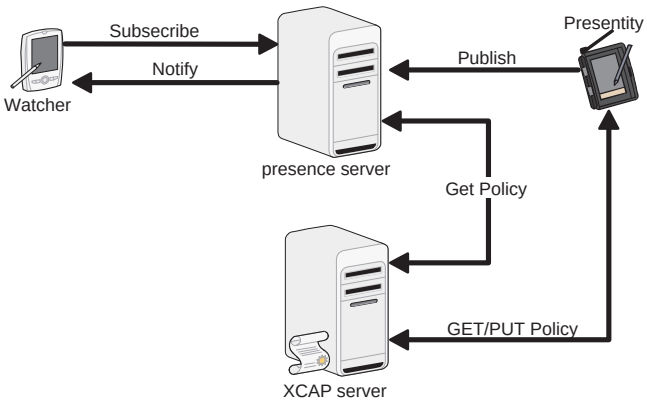

Fig. 1. Architecture for presence service

In order to generate interoperability test cases automatically and to minimize the test size, a number of works have been done [15][16][17]. Hao and *et al*. [15] proposed an algorithm to generate a minimum number of test cases using reachability graph and next transition graphs where the system behaviors are modeled by Extended Finite State Machines (EFSMs). In many cases, however, we may have state-explosion problem when test cases are generated from communicating systems where each system is modeled by an EFSM. There have been a number works to solve the state-explosion problem during interoperability test case generation [16]. Desmoulin and Viho [17] proposed a method that generates interoperability test cases avoiding state-explosion problem by using test purposes. In [17], however, it is necessary to describe test purposes in Input Output Labeled Transitions Systems (IOLTS) which may limit the applicability of the proposed method.

III. Presence service

Presence service provides the ability for the home network to manage presence information of a user device as well as information of services. Users of presence service such as watchers and presentities can be located either in their home networks or in visited networks. In 3GPP, a presence service is defined as the capability to support management of presence information between watchers and presentities, in order to enable application and services to make use of presence information. The presence information is defined as a set of attributes characterizing current properties of presentities including status and other optional attributes such as communication address [2]. Figure 1 shows a simplified architecture for presence service.

As shown in figure 1, we have four entities; a presence server, an XCAP server, presentities and watchers. A presence server communicates with two clients, presentities and watchers using presence protocol. There are a number presence protocols such as SIMPLE and XMPP (Extensible Messaging and Presence Protocol) [18] defined by IETF, and Wireless village which is established by Ericsson, Motorola, and Nokia. Since we consider the presence service on IMS platform,

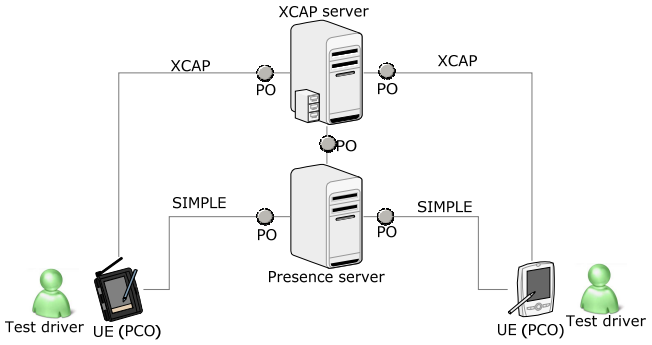

Fig. 2. Generic test architecture

SIMPLE is used as a presence protocol.

Presentities provide presence information to a presence server using *Publish* message. Watchers subscribe to a presence server for presence information of presentities and then receive presence information using *Notify* messages. There are two kinds of watchers, fetchers and subscribers. A fetcher simply requests the current presence information of presentity from the presence server. A subscriber requests notification from the presence server whenever new presence information is available. A special kind of fetcher, called poller, fetches information on a regular basis. An XCAP server is a generic server that provides the ability to query, modify or delete data stored in an XML format. An XCAP server communicates with presentities using XCAP [19] and with a presence server using XML-RPC [20] where both are HTTP-based protocols. In presence service, presence lists are manipulated by watchers and authorization policies by presentities, which are stored in an XCAP server in XML format.

IV. Test architecture and deployment of testbed

A. Interoperability test architecture

In our experiments, a presence server, an XCAP server, and clients such as presentities and watchers are considered for interoperability testing. Therefore, test cases related to Resource List Server (RLS) are not considered in this paper. In the test cases developed by OMA [3][4][5], it is assumed that PCO's (Points of Control and Observation) are on UEs (User Equipments)' applications. In order to facilitate the analysis of test results, e.g. to identify the location of the problem, we added a PO (Point of Observation) for each interface using Wireshark [21]. Figure 2 shows a generic test architecture used in this paper.

Recall that watchers and presentities can be located either in their home networks or in visited networks. Since the test architecture shown in figure 2 is an abstract one, the locations of users do not make any change in the test architecture. For the same reason, test cases developed by OMA are independent of users location.

B. Deployment of the IMS platforms

We deployed the open source IMS platform developed by Fraunhofer FOKUS, called OpenIMS [8]. The components provided by OpenIMS include CSCFs (Call Session Control Functions), HSS (Home Subscriber Server), MG (Media Gateway), MRF (Media Resource Function), Application Servers, and SIP2IMS gateway. CSCFs (P-CSCF, I-CSCF, and S-CSCF) of the OpenIMS are built upon the SIP Express Router (SER) [22] where SER is a high-performance open SIP server licensed under the open-source GNU license. Each x-CSCF component can be configured to have its own IP address and port number. In our experiments, we installed CSCFs and HSS in a PC and the same IP address was assigned for each component with different port numbers.

Since watchers and presentities can be located in visited networks, it is necessary to build at least two network domains. For the case of two network domains, which will be called inter-domain case, we can install each one in a different PC with its own domain name and CSCFs are configured so that it can handle requests either from clients or from other networks[1].

C. Deployment of the servers and clients

We found two open source presence servers: one based on SER and the other on OpenSER. Since the most recent stable version of SER (version 0.9.6) does not support XML-RPC management interface which is necessary for communication with XCAP servers, the experiments were carried out using the presence server based on OpenSER. We configured OpenSER to be a presence server and then created service profiles for the users in HSS including subscription information for presence service and the address of the presence server. We installed OpenXCAP [10] developed by AG projects.

In our experiments, we used two IMS clients from different vendors, which is also mentioned as a mandatory requirement by OMA [3]. We found two open source IMS clients, UCT clients [11] and IMS Communicator [12], that support presence service using SIMPLE presence protocol.

V. Experimental results

The test cases provided by OMA can be categorized into three parts: Presence SIMPLE, Presence XDM (XML Document Management), and RLS XDM. Presence SIMPLE defines interworking scenarios among presence server, XCAP server, watchers, and presentities. Presence XDM and RLS XDM define functionalities related to handling databases of servers such as XCAP server and RLS. In this paper, we performed experiments using test cases of Presence SIMPLE and Presence XDM. Due to non-supported functionalities by clients such as handling of presence composition rules, subscription/notification filtering, and partial presence information publishing, we tested 15 test cases out of 26 for Presence SIMPLE. For the case of Presence XDM, we tested 3 test cases, each one for creation, removal, and modification of authorization rules.

[1]We tried to build two IMS platforms from different vendors but unfortunately, we could not find any other open source so we installed OpenIMS for both PCs

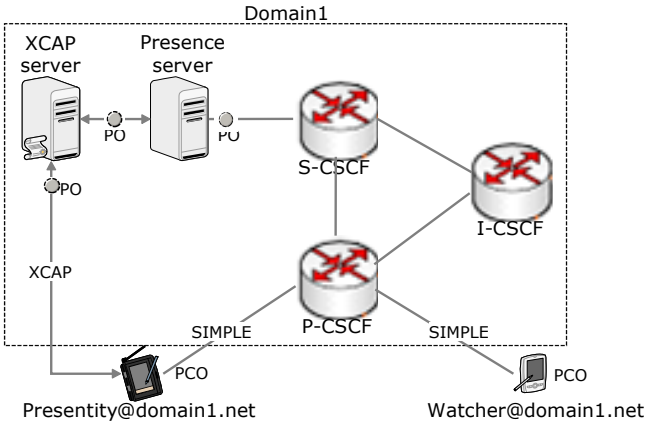

Fig. 3. Intra-domain case

As mentioned in section IV, watchers and presentities can be either in their home networks or in visited networks. In this paper, we tested the case where watchers and presentities have the same home network and both are in the home network, which will be called intra-domain case. 18 test cases were applied and the results were analyzed. For other cases, e.g. presentities are in visited networks, which we will call inter-domain cases, we did not perform interoperability testing since we are using the same implementations of IMS platform, presence server, and XCAP server with the same optional attributes of presence information. Therefore, for inter-domain cases, in this paper, we explain possible testbeds and discuss briefly the differences with intra-domain case.

A. Intra-domain case

In this case, two clients are considered to be in the same domain; e.g. domain1 in figure 3. As mentioned above, we divided our tests cases into two parts; the testing of presence SIMPLE (presence features) and the testing of presence XDM (functionalities related to handling databases of XCAP server).

1) Presence SIMPLE:

Test 1) *Publication of Presence information*: Verify that presence information published by a UE will be received by another UE, which subscribes for that information.
Test 2) *Publication of Presence information, publish modification*: Verify that presence information modified by a UE will be displayed accordingly in another UE, which subscribes for that information.
Test 3) *Publication of Presence information, subscription removal*: A UE, acting as a Watcher terminates its subscriptions, and another UE, the presence source, updates the presence information.
Test 4) *Publication of Presence information, subscription refresh*: Verify that Presence Server keeps sending presence information to a UE, acting as a watcher, after subscription refresh.
Test 5) *Notification of Presence information from multiple Presentities*: Verify that a Presence Server can store and manage presence information coming from multiple UEs, acting as Presence Sources and related to several Users, and correctly notify one UE, acting as a Watcher the presence information.
Test 6) *Distribution Policy (presence content rules I)*: Verify that a User is able to define policies so that different presence information can be sent to different Users, acting as Watchers.
Test 7) *Distribution Policy (presence content rules II)*: Verify that a User is able to define policies so that the same presence information elements but with different values can be sent to different Users, acting as Watchers.
Test 8) *Publication of Presence information, subscription Poll Request*: Verify that a UE successfully publishes and retrieves presence information by polling.
Test 9) *Default policy*: Verify that a User is able to define policies so that defined presence information can be sent to unspecified Users (not known in the Presence Rules document), acting as Watchers.
Test 10) *Authorization management for groups*: Verify that a presence server can handle the presence rules document for groups of Watchers stored in the XCAP server.
Test 11) *Combining permissions on an ongoing subscription*: Verify that a Presence Server can handle changes for the Presence Rules document for Watchers (individual Watchers or groups) stored in the XCAP server.
Test 12) *Publication of Presence information, watcher is blocked*: Verify that User1 successfully publishes presence information. User2 will not be able to subscribe to the presence information when blocked by User1.
Test 13) *Publication of Presence information, watcher is politely blocked*: User2 will be able to subscribe and receive notifications, but presence information will not be revealed, since the user is politely blocked.
Test 14) *Reactive authorization for a specific group*: Verify that a presentity can authorize a group of watchers to subscribe to his/her presence information when the request from that watcher arrives (Reactive authorization).
Test 15) *Combination of presence elements from different presence sources*: Verify that a Presence Server supports the combination of different presence information elements of a particular user coming from different UEs, acting as presence sources.

2) Presence XDM:

Test 16) *Presence XDMS document creation, retrieval and validation*: Verify that the user can create and retrieve an XML document from the XCAP server.
Test 17) *Presence XDMS document and element modification, retrieval and validation*: Verify that the UE can modify and retrieve XML elements and documents from the XCAP server.
Test 18) *Presence XDMS element deletion, retrieval and validation*: Verify that the UE can delete XML elements from the XCAP server.

B. Test Results

As shown in Table I, we applied 18 test cases and repeated 10 times for each. Among 180 test applications we obtained

TABLE I
TEST RESULTS

# of test cases	# of sessions	# of passes	# of fails
18	10	148	32

32 fail verdicts. We analyzed the test results in detail and three problems were found in presence server and clients as follows:

1) Handling of a Subscribe message in presence server when the Expires header has the value zero: When the presence module of OpenSER receives a *Subscribe* message from a watcher where the *Expires* header has the value zero (Test 3), it should i) remove the watcher from active watcher table and ii) send a notification of the presence information of the presentity to the watcher. In case of OpenSER, a notification is sent after the reception of the *Subscribe* message, but it does not remove the watcher from active watcher table. Therefore, the watcher continues to receive notifications from the presence server and it respond with an *Error* message (subscription does not exist).

2) Handling of combined presence information in a client: When several user equipments of the same client send presence information to a presence server (Test 15), the presence server combines the received presence information in an XML document. In such a case, watchers can receive presence information of multiple equipments. Although the presence module of OpenSER sends a notification message with combined presence information, neither UCT IMS client nor IMS Communicator can handle this information. Both clients display the presence information of the user equipment that was published the last.

3) Publishing of information when an equipment is powered off: When an equipment is powered off (Test 1), it should send a *Publish* message to a presence server containing the parameter "status=close"[2]. In case of IMS Communicator, sometimes (two cases out of ten in our experiments), it does not send the *Publish* message to a presence server so the equipment is supposed to be online by watchers.

C. Inter-domain case

In this section, we explain testbeds for inter-domain cases and discuss briefly the differences with intra-domain case. According to test cases, we can have various kinds of interactions between network components. For simplicity, the following two cases are considered:

Case I: It is assumed that a watcher and a presentity have different home networks and they are in their home networks as shown in figure 4. In this case, it is important to test the interaction between a presence server and a watcher since the watcher should be able to subscribe and get notifications from the presence server in different domain, e.g. domain1 in figure 4. For the test cases where all the interactions are made within domain1 such as interaction between a presentity and the XCAP server, there is no difference with the intra-domain case.

[2]Although this is not specifically described in the test case, we think that it is reasonable to send the *Publish* message when a UE is powered off.

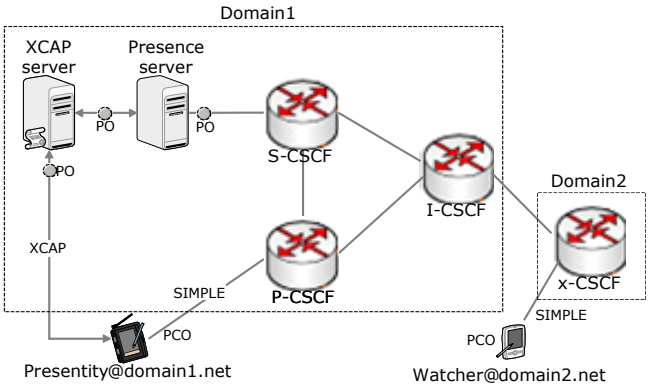

Fig. 4. Inter-domain Case I: two UEs in their home networks

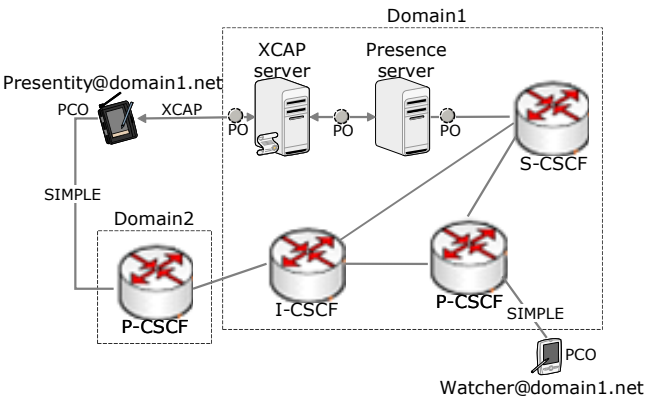

Fig. 5. Inter-domain Case II: presentity in visited network

Case II: It is assumed that a watcher and a presentity have the same home network and the presentity is in visited network as shown in figure 5. In this case, it is important to test the interaction between a presence server and a presentity since the presentity should be able to publish presence information to the presence server in different domain, e.g. domain1 in figure 5.

VI. CONCLUSION

In this paper, we performed experiments on interoperability testing of presence service on IMS platform. Open source implementations such as an IMS platform, a presence server, an XCAP server, and IMS clients were deployed to establish testbed for interoperability testing of presence service. Test cases for SIMPLE presence protocol developed by OMA were applied to our testbed and test results were analyzed in detail. Due to the non-supported functionalities by clients such as handling of presence composition rules, subscription/notification filtering, and partial presence information publishing, it was not possible to test all test cases defined by OMA. We obtained fail verdicts for some test cases because of problems in presence servers and IMS clients.

In our experiments, we covered the basic functionalities of the presence service and the interaction between a client and an XCAP server for intra-domain case. And then we explained the testbeds for inter-domain cases and briefly discussed the

differences with intra-domain case. The next step can be performing the interoperability testing of presence service in inter-domain cases. Also we are working on generation of test cases for interoperability testing of services which can provide scalability.

REFERENCES

[1] *Methods for Testing and Specification (MTS); Internet Protocol Testing (IPT); Generic approach to interoperability testing*, ETSI Std. EG 202 237, 2007.

[2] *Presence Service; Architecture and functional description*, 3GPP Std. TS 23.141, 2007.

[3] *Enabler Test Specification for Presence SIMPLE Interoperability*, Version 1.1, Open Mobile Alliance (OMA) Std., 2008.

[4] *Enabler Test Specification for Presence XDM Interoperability*, Version 1.1, Open Mobile Alliance (OMA) Std., 2008.

[5] *Enabler Test Specification for Presence RLS XDM Interoperability*, Version 1.1, Open Mobile Alliance (OMA) Std., 2008.

[6] "Enabler test report presence SIMPLE," OMA Test Fest, 2006.

[7] Wireless Village. [Online]. Available: http://www.openmobilealliance.org/tech/affiliates/wv/wvindex.html

[8] Fraunhofer FOKUS. Open IMS Playground. [Online]. Available: http://www.fokus.fraunhofer.de/en/fokus_testbeds/open_ims_playground/

[9] OpenSIPS Project. [Online]. Available: http://www.opensips.org/

[10] OpenXCAP. [Online]. Available: http://openxcap.org/

[11] UCT Client. [Online]. Available: http://uctimsclient.berlios.de/

[12] IMS Communicator. [Online]. Available: http://imscommunicator.berlios.de/

[13] J. Rosenberg, "SIMPLE made simple: An overview of the IETF specifications for instant messaging and presence using the session initiation protocol," Internet draft, 2008. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-simple-simple-04.txt

[14] OMA TestFest. [Online]. Available: http://www.openmobilealliance.org/TestFests/overview.aspx

[15] R. Hao, D. Lee, R. K. Sinha, and N. Griffeth, "Integrated system interoperability testing with applications to voip," *IEEE/ACM Trans. Netw.*, vol. 12, no. 5, pp. 823–836, 2004.

[16] H.-M. Noh, J.-H. Lee, C.-J. Yoo, and O.-B. Chang, "Behavior modeling technique based on efsm for interoperability testing," in *Proceedings of the International Conference on Computational Science and Its Applications (ICCSA 2005)*, 2005, pp. 878–885.

[17] A. Desmoulin and C. Viho, "Interoperability test generation: formal definitions and algorithm," in *Huitieme Colloque Africain sur la Recheche en Informatique (CARI'06)*, Cotonou, Benin, Nov. 2006.

[18] P. Saint-Andre, "Extensible messaging and presence protocol (XMPP): Instant messaging and presence," RFC 3921, 2004.

[19] J. Rosenberg, "The extensible markup language (XML) configuration access protocol (XCAP)," RFC 4825, 2007.

[20] D. Winer, "XML-RPC specification." [Online]. Available: http://www.xmlrpc.com/spec

[21] Wireshark. [Online]. Available: http://www.wireshark.org/

[22] SIP Express Router. [Online]. Available: http://www.iptel.org/ser/