**Contemporary Engineering Sciences, Vol. 7, 2014, no. 11, 531 - 542**
**HIKARI Ltd, www.m-hikari.com**
**http://dx.doi.org/10.12988/ces.2014.4440**

# Communication Models of Presence Service

**David Petras**

Liptovsky Mikulas, Slovakia

**Ivan Baronak**

Bratislava, Slovakia



## Abstract

This paper describes the presence service which is located in the IP multimedia subsystem (IMS). This service allows creation of many applications for different groups of people. This paper describes the number of transmitted messages in the network intended for users who track statuses of other users. Apart from the number of the messages it also deals with the composition of these messages. Here we describe the difference between these models of communication. Specifically we talk about the push model, the pull model and the combined model. We compare these models and choose which to use and when.

**Keywords**: IP multimedia subsystem, Presence service

## 1 Introduction

Originally, each service or group of services had its own network for its use. The idea of the Next Generation Networks (NGN) came with the development of technology. This network would be shared by all services. IMS is one of the forms of NGN [1]. IMS joins networks with packet switching and circuit switching. The signalization protocol in IMS is Session Initiation Protocol *(*SIP) [2-3]. It transmits voice through Voice over Internet Protocol (VoIP) [4-6]. It offers a number of advantages. Integration of new services into this system does not require significant interference with the structure of the network. Sufficient quality of these services is guaranteed thank to the implementation of QoS parameters.

The presence service should inform others about the user´s status and enable the user to see statuses of others [7]. It is transmitted through the extension of SIP protocol for instant messaging and the presence of SIMPLE [8]. This protocol allows the transmission of messages over the network without changes when the service is deployed. Messages are created as XML documents. They have a format PIDF [9] and its various extensions such as Rich Presence Information Data Format RPID [10].

## 2 IP Multimedia Subsystem

IMS consists of three layers: transport, control and application. The transport layer is the lowest. This layer has two roles. The first role is to secure the access of devices from different types of networks (GPRS, UMTS, IP, PSTN...) through gateways. The second role is to transfer messages from the user to the control layer, or from one user to another user through IP data network. The control layer is the core of the IMS network. It controls the communication and creates connections among users. It directs messages through three Call Session Control Function (CSCF) servers. P-CSCF is an entry point to IMS. I-CSCF is used at registration and communication between two IMS networks. S-CSCF is a central point of direction. It communicates with application servers. Home Subscriber Server (HSS) is a database where user profiles and data for the services are stored. The Subscriber Location Function (SLF) selects the database from several HSSs. The application layer provides services. There are three types of servers. SIP application server for applications using the SIP protocol. OSA (Open Service Access) application server is independent from the used protocol thank to Application Programming Interface (API) between the server and the control layer. CAMEL is an application server used for applications created in networks through circuit connections [6].

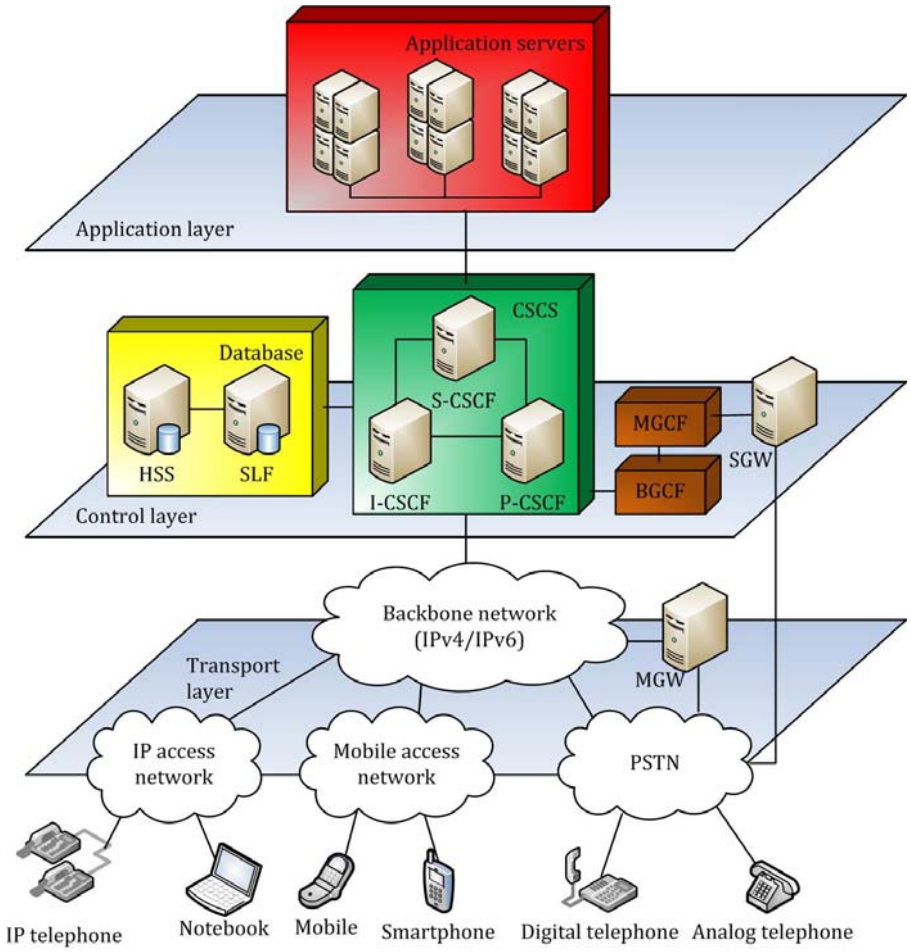


Fig 1: Architecture of IMS

## 3. Presence architecture

Presence architecture is shown in figure 2. It has three levels: agents, entities and a server. Agents collect information from various sources. These are various programmes on network elements. Presence network agent collects information from the network elements. Presence user agent collects information from the user´s device. Presence external agent collects information from other networks. Watcher presence agent provides information for the watcher.

Entities are characterized by the fact that they can work with SIP messages (UE, S-CSCF, AS). Entities are divided into two types. Presentity (Presence entity) provides information about itself and the Watcher observes the status of the others [14]. The presence server gathers and sends information to users and stores them in XML documents. One part of the server receives the messages and assigns them

to the correct user. The second part creates lists of users for the watcher and sends him their status together. The third part (XML document manager server – XDMS) supports other parts of the Presence server. For example it knows which watcher is authorized to observe the presentity. This application server is designed so that it can control a number of messages. One of the possibilities is a periodic sending of messages. If there are more messages than the server can send, it puts them into a waiting queue or if the waiting queue is full then the server deletes the messages [15].

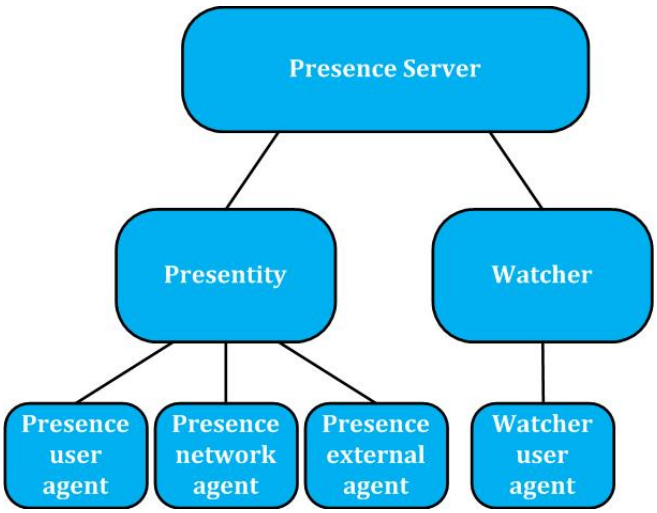


Fig 2: Presence service architecture

## 4. Messages

The presence service uses three types of SIP requests for communication – Publish, Subscribe and Notify. In the request the Publish presentity sends information about its status. Subscribe request is sent by the watcher, if he requests information about the status of presentities. There is a filter in this request where the watcher declares which presentities he wants to track. There can also be an information subscription period. The second possibility is that the presentity sends a Subscribe request to find out about the tracker. The server announces the status of presentities to the watcher through a Notify request. All requests are confirmed by SIP replies (e.g. 200 – OK, 401 – Unauthorized).

Information in the subject of the request is organized into groups called "tuple". The arrangement of the information is described in the Presence data model. This model consists of four parts – URI of presence, group "people", group "services" and group "devices". URI of presence together with SIP URI identifies the user of the service and it is unique for each participant. Basic statistic (gender) and dynamic (mood) information about the user is stated in the group "people". The

group “services” contains information about the specific service, e.g. status – online/offline. The group “devices” informs about classification options of the users (video call support). The user in IMS has more classifications, that is why there are more groups [16].

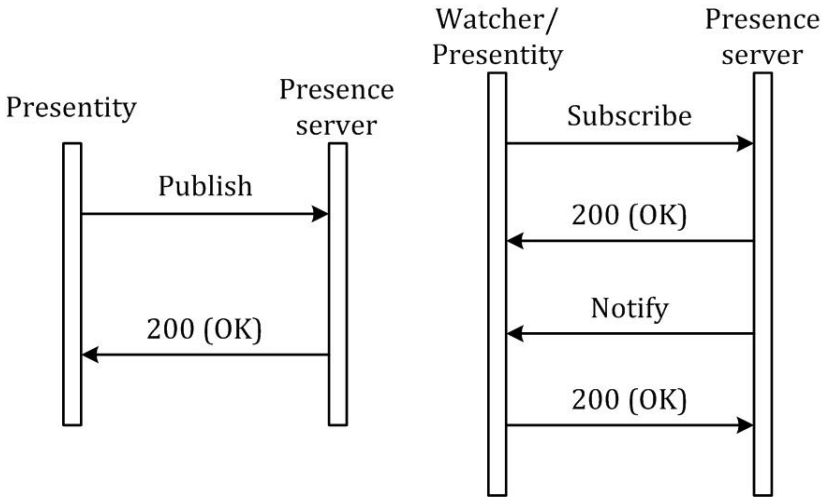


Fig 3: Communication

## 5. Push model and pull model

The Push model is a model where the watcher logs in the server to get information. This way the watcher is constantly informed about the current status of presentities. The Pull model represents a model where the watcher only detects the current status and ignores the changes. Choosing one of these models represents the most significant difference in the character of the service. This choice determines the number of transmitted messages and informedness of the watchers [15].

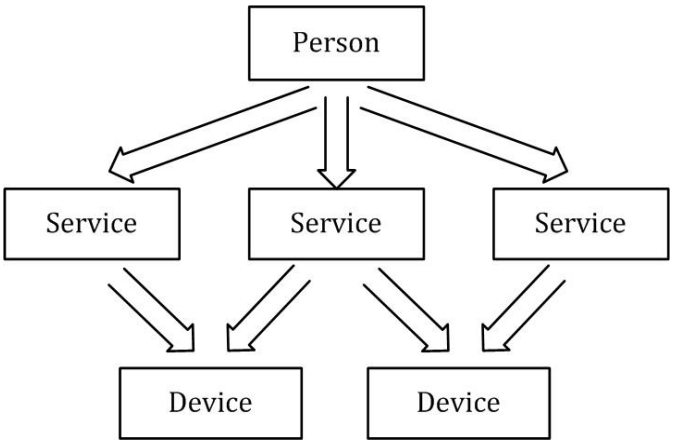


Fig 4: Presence service data model [16]

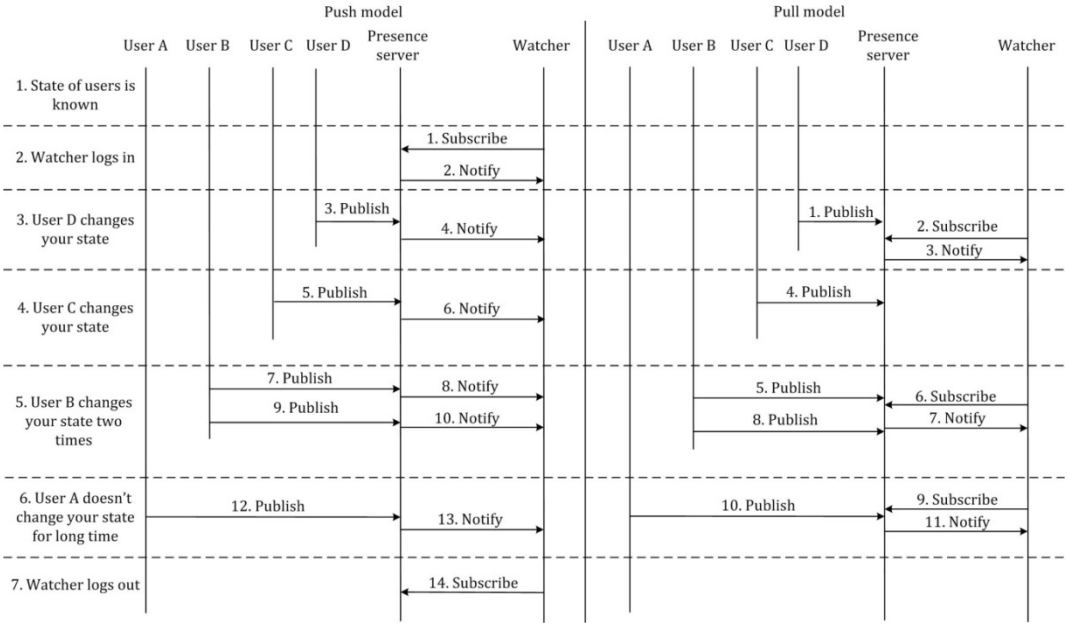


Fig 5: Comparison of transmitted messages

Transmitted messages of individual models are shown in figure 5. In the first step it is assumed that all users have informed the server about their status. In the second step in the push model, the watcher connects to the service – for instance automatically when turning on a mobile phone. The watcher sends messages about the initiation of the subscription and the server sends him the status of the presentities. There is nothing sent in the Pull model, because the user has not started using that particular service. In the third step the user D sends information about the change of his status to the server, also the watcher gets this information in the pull model. At the same time the watcher wants to call one of the users. No message to the presence service has to be sent in the push model, because the watcher is informed about the status of the users. In the request model the user first finds out about the status of presentities and only then starts the calling. Only the user C changes his status in the fourth step and only the watcher in the pull model is informed about this. In the fifth step user B changes his status twice. The watcher in the model of request has just checked the status of these two changes and is not informed about the current status of user B. In the sixth step user A changes the status and watchers in both models are informed about this. In the seventh step the watcher logs out of the service. He quitted subscription in the push model.

There are shortcomings in the model of request in the period when presentity changes its status after the server already sends the Notify message and the watcher still checks the status of presentities. This shortcoming can be eliminated when the watcher, after receiving of the status of the presentities, follows changes of the status of presentities. Due to this the number of messages is increased, but the user can be sure, that the status of presentities is current even after a longer

time when the first message was sent.

The choice of the model does not influence the number of *publish* messages, that is why this paper does not deal with these messages in detail. These messages are described in [17]. $Push_{subscribe}$ messages are created when the watcher wants to log in ($sub_{initial}$), extend ($sub_{refresh}$) and log out ($sub_{terminal}$) the information subscription from the server. $Pull_{subscribe}$ messages are created when the watcher wants to find out about the current status of the entities through a $sub_{initial}$ message [18]. In the joined model the $push\_pull_{subscribe}$ messages will be sent at the initiation and refreshing of the information subscription from the server. Notify messages are a response to either the Publish message ($notify_{pub}$) or Subscribe message ($notify_{sub}$). *Watchers* represent an average number of watchers per one presentity. The number of $notify_{pub}$ messages is percentually decreased in the joined model, depending on the watching parameter. The *watching* parameter is set as the ratio of the average tracking of presentities and total time.

$$push_{subscribe} = sub_{initial} + sub_{terminal} + sub_{refresh} \quad (1)$$

$$pull_{subscribe} = sub_{initial} \quad (2)$$

$$push\_pull_{subscribe} = sub_{initial} + sub_{refresh} \quad (3)$$

$$push_{notify} = notify_{pub} + notify_{sub} \quad (4)$$

$$pull_{notify} = notify_{sub} \quad (5)$$

$$push\_pull_{notify} = notify_{sub} + notify_{pub} . watching \quad (6)$$

$$notify_{sub} = sub_{initial} \quad (7)$$

$$notify_{pub} = watchiers . publish \quad (8)$$

To set the number of transmitted data is it necessary to know also the average size of individual messages, not only the number of messages. $Subscribe_{len}$ is the average size of Subscribe messages. The average size of Notify messages differs according to the type of the message. $Notify_{sub}$ messages contain information about status of all users and that is why the average size of $notify_{sub\_len}$ is bigger than $notify_{pub\_len}$, which is an average size of $notify_{pub}$ messages.

## 6. Model comparison

Picture 6 compares ratios of transmitted messages of individual models. Graphs a, d, g describe the push model. The pull model is described in graphs b, d, h. The remaining three graphs c, e, i show the combined model. *Watching* in the graphs for the combined model is 0,1. Parameters of the individual graphs are as follows:

a, b, c - *publish* 1000 *watchers* 1-200 *subscribe* 600

d, e, f - *publish* 1-1000 *watchers* 50 *subscribe* 600
e, h, i - *publish* 1000 *watchers* 50 *subscribe* 1-200

It is clear from the graphs that Notify messages are significantly more dominant in the push model. The number of Notify messages was decreased in the pull model. The number of Notify messages is only higher in case when watchers check the status of presentities more often than presentities change their status. Notify messages prevail in the combined model, although their number is almost ten times lower than in the push model.

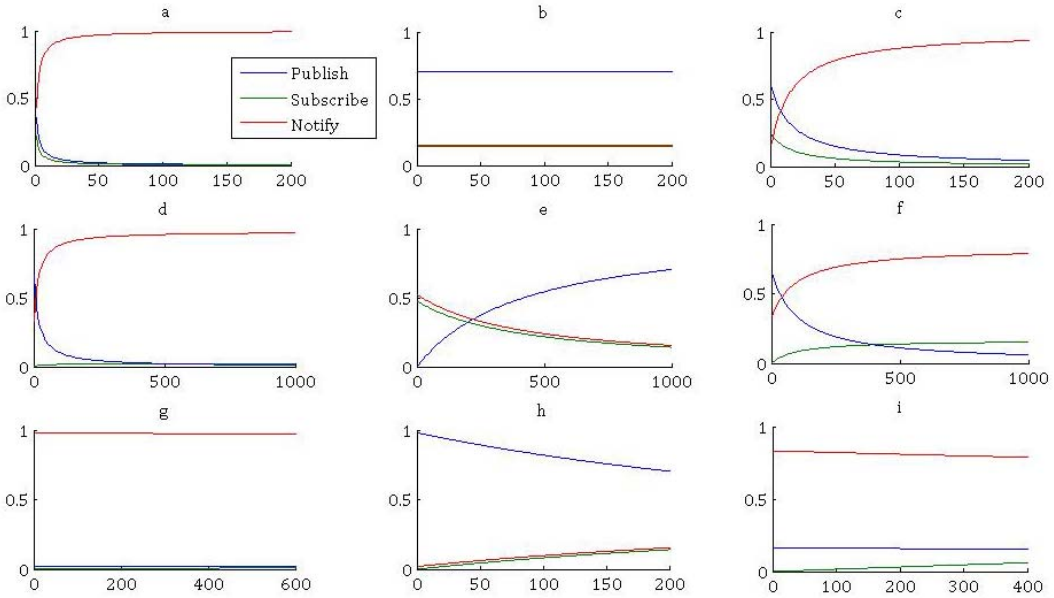


Fig 6: Ratio of transmitted messages

When comparing the models it is important how often the watchers will check the status of presentities. In the pull model the watcher sends a Subscribe message every time he checks the status. He receives a reply to this message, which contains the number of all users. In the push model Subscribe is only sent upon initiation and termination of information subscription from the server. The watcher receives a message with all statuses upon logging in and then receives messages only when the status is changed. Messages about the status change are much smaller than the first Notify message. Picture 7 shows a relation of the number of transmitted data and the number of viewing of the status of entities in certain time. Parameters for this graph are as follows:

$subscribe_{len}$= 3 kB
$notify_{sub\_len}$ = 60 kB
$notify_{pub\_len}$ = 2 kB
$push_{subscribe}$ = 450

$pull_{subscribe}$ = 100- 300
$notify_{sub}$ = 150
$notify_{pub}$ = 1000

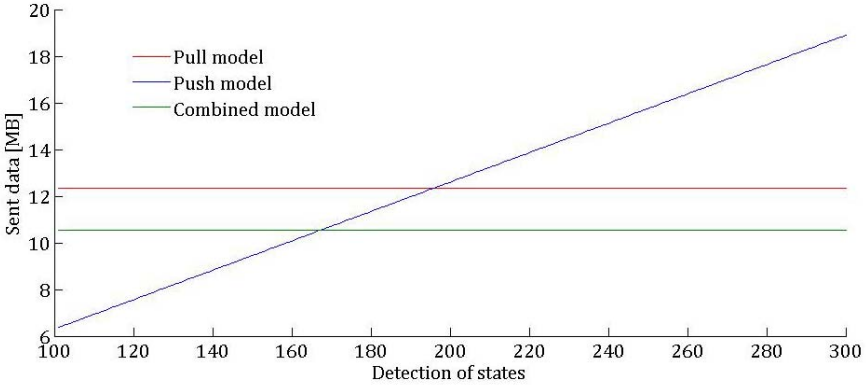


Fig 7: Ratio of transmitted messages

The selection of the model can be set according to relation 11, where *E* sets the ratio among the number of transmitted messages in various models of communication. Pull model will be more effective if *E*<1 according to this relation, other times it will be better to select a different model.

$$push_{messages} = push_{subscribe}.subscribe_{len} + notify_{sub}.notify_{sub\_len} + notify_{pub}.notify_{pub\_len} \quad (9)$$

$$push_{messages} = push_{subscribe}.subscribe_{len} + notify_{sub}.notify_{sub\_len} + notify_{pub}.notify_{pub\_len} \quad (10)$$

$$E = \frac{push_{messages}}{pull_{messages}} \quad (11)$$

The choice between the push and the combined model depends on the time of watching the status of presentities by watchers. If the average checking time is for instance 55 seconds, the combined model with the period of sending messages $sub_{refresh}$ is one minute. Then even if the watcher does not check the status of presentities, a number of redundant $notify_{pub}$ messages is not sent. When the time of watching is long, for instance 3 hours, it is better to use the push model with a longer period of $sub_{refresh}$ messages, bearing in mind that when the status subscription is terminated, a $sub_{terminal}$ message is sent. Table 1 shows when it is appropriate to use the given model.

Table 1. Model comparison

| Model | Time of watching | Detection of states |
|---|---|---|
| Push model | Long | Often |
| Pull model | Instant state | Sometimes |
| Combined model | Short | Often |

## 7. Conclusion

Presence service is one of the key services in IMS. It allows creation of a huge number of applications which can share information. The service itself should be the source of a great number of messages, that is why it is important to know the character of the service before its usage in a network. Three main questions arise when selecting an appropriate communication model. How long and how often will the watcher check the status of presentities and how many messages can be transmitted over the network. If the watchers only need to check the status of presentities from time to time and they immediately use it – it is appropriate to use the pull model. If aside from the occasional checking of the status, watchers also check the status for a longer period of time – it is necessary to use the combined model. The push model is appropriate when checking is permanent. The push model and the combined model are appropriate when the checking of the status of presentities is frequent. The pull model is appropriate with occasional checking. In the network with unlimited resources it is necessary to properly identify the behaviour of watchers and then select an appropriate model with the help of the relation 9 and figure 7. The key question here is how often the watcher will check the status of presentities. If the network does not meet the parameters required by the service in any way, it is necessary to use other possibilities to decrease the number of transmitted messages. This can be achieved for instance by decreasing the number of watchers with the help of a filter, connecting Notify messages or compression of messages.

**Acknowledgements.** This article was created with the support of the Ministry of Education, Science, Research and Sport of the Slovak Republic within the Research and Development Operational Programme for the projects "Centre of Excellence for SMART Technologies, System and Services II", ITMS 26240120029 and "University Science Park of STU Bratislava", ITMS 26240220084, co-funded by the European Regional Development Fund.

## References

[1] N. Wilkinson, Next Generation Network Services – Technologies and Strategies, Wiley, New York, 2002.
[2] F. Rezac, M. Voznak, K. Tomala, J. Rozhon, and J. Vychodil, Security analysis system to detect threats on a SIP VoIP infrastructure elements, Advances in Electrical and Electronic Engineering, Volume 9, Issue 5, 2011, Pages 225-232.
[3] M. Voznak, and J. Safarik, DoS attacks targeting SIP server and improvements of robustness, International Journal of Mathematics and Computers in Simulation, Volume 6, Issue 1, 2012, Pages 177-184.
[4] M. Voznak, and F. Rezac, Web-based IP telephony penetration system evaluating level of protection from attacks and threats, WSEAS Transactions on Communications, Volume 10, Issue 2, February 2011, Pages 66-76.
[5] J. Rozhon, P. Blaha, M. Voznak, and J. Skapa, The weather impact on speech quality in GSM networks, Communications in Computer and Information Science, Volume 291 CCIS, 2012, Pages 360-369.
[6] J. Rozhon, and M. Voznak, Development of a speech quality monitoring tool based on ITU-T P.862, 34th International Conference on Telecommunications and Signal Processing, TSP 2011 - Proceedings, art. no. 6043771, pp. 62-66.
[7] M. Poikselka, G.Mayer, H. Khartabil, and A.Niemi, The IMS: IP Multimedia Concepts and Services in the Mobile Domain, Wiley, New York, 2004.
[8] S. Leggio, SIP for instant messaging and presence leveraging extensions, 2005.
[9] H. Sugano, S. Fujimoto, G. Klyne, A. Bateman, W. Carr, and J. Peterson, RFC 3863: presence information data format (PIDF), 2004.
[10] H. Schulzrinne, U. Columbia, V. Gurbani, P. Kyzivat, and J. Rosenberg, RFC 4480: RPID: rich presence extensions to the presence information data format (PIDF), 2006.
[11] K. Al-Begain, C. Balakrishna, L. A. Galindo, and D. M. Fernández, IMS: A Development and Deployment Perspective, Wiley, New York, 2009.
[12] S. Chakraborty, T. Frankkila, J. Peisa, and P. Synnegren, IMS Multimeda Telephony over Cellular Systems, Wiley, New York, 2007.
[13] H. Khlifi, and J. C. Gregoire, IMS application servers, Roles, requirements, and implementation technologies, IEEE Internet Comput., vol. 12, no. 3, pp. 40–51, 2008.
[14] M. Day, J. Rosenberg, and H. Sugano, RFC 2778: a model for presence and instant messaging, 2000.
[15] M. Wuthnow, M. Stafford, and J. Shih, IMS: A New Model for Blending Applications, Taylor & Francis, Florida, 2010.
[16] G. Camarillo and M. Garcia-Martin, The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds, Wiley, New York, 2nd edition, 2006.
[17] D. Petráš, I. Baroňák, and E. Chromý, "Presence Service in IMS," The Scientific World Journal, vol. 2013, Article ID 606790, 8 pages, 2013, http://dx.doi.org/10.1155/2013/606790.

[18] C. Chi, R. Hao, D. Wang, and Z. Cao, IMS presence server: traffic analysis & performance modelling, Proceedings of the 16th IEEE International Conference on Network Protocols (ICNP'08), pp. 63–72, 2008.

**Received: April 30, 2014**