DESIGN AND SOFTWARE ARCHITECTURE OF SIP SERVER FOR OVERLOAD CONTROL SIMULATION

Pavel O. Abaev (speaker)
Yuliya V. Gaidamaka
Konstantin E. Samouylov
Telecommunication Systems Department
Peoples' Friendship University of Russia
Miklukho-Maklaya str., 6,
117198, Moscow, Russia
Email: pabaev@sci.pfu.edu.ru,
ygaidamaka@sci.pfu.edu.ru
ksam@sci.pfu.edu.ru

Sergey Ya. Shorgin
Institute of Informatics Problems of RAS
Vavilova, 44-1,
119333, Moscow, Russia
Email: sshorgin@ipiran.ru

KEYWORDS

Signalling network, SIP, hop-by-hop overload control, hysteretic load control, simulation tool.

ABSTRACT

The rapid development of services provided on SIP networks not only define the necessity of creating new equipment and standards but also requires the development of new methods and programming software tools for modeling and analyzing the effectiveness of the overload control mechanisms in SIP-server networks. The modeling process utilizes mathematical and simulation models as well as simulators. Only simulation tools make it possible to solve problems related to the analysis and optimization of the control parameters. The most appropriate modeler is the simulators reflecting the protocols and functions which are fully or partially built into the original system. Over the last few years, IETF working group SOC (SIP Overload Control) has been actively conducting research aimed at developing an effective mechanism for server overload control in SIP networks. At present, there are no simulators for modeling the work of SIP servers in overload conditions with an application of mechanisms which are currently under development by SOC. A model for a SIP server simulator and approaches to its programming implementation are proposed in the paper.

INTRODUCTION

SIP is an application-layer signaling protocol for creating, modifying, and terminating sessions with one or more participants. In November 2000, SIP was accepted as a 3GPP signaling protocol and main protocol of the IMS architecture. In 2002, recommendation (RFC 3261, 2002) which determines the current protocol form was accepted. The rapid development of the market for services based on the SIP protocol and the growing user needs have revealed a number of shortcomings in the protocol, specifically, in the base overload control mechanism (mechanism 503). In 2009, Rosenberg, one of the protocol designers, demonstrated in (RFC 5390, 2008) the protocols' main shortcomings in regard to overload prevention and formulated the main requirements toward the future overload control mechanism. In mid-2010, the SOC working group was created within the IETF Committee. Its work aims at creating overload prevention mechanisms. The first result of their work was the document (RFC 6357, 2011) which was permanently accepted in August 2011. The document provides a discussion of the available types of overload control mechanisms – local, hop-by-hop, and end-to-end, a classification of SIP networks, and presents the overall architecture of overload-control systems. The SOC group's work focuses on developing two hop-by-hop schemes for overload control as this type of mechanisms has a number of indisputable advantages over the other two types (RFC 6357, 2011; IETF draft SIP Rate Overload Control, 2013). At present, two overload control schemes have been proposed – one with flow sifting on the sender side (LBOC, Loss-based overload control) and one with restricting the flow rate of signaling messages (RBOC, Rate-based overload control). However, only the basic principles were described in SOCs' documents and methods for calculation of the control parameters were not specified. The control parameters can be determined based on analysis of mathematical models or as the results of simulation modeling. As the processes going on in the SIP networks are difficult to describe mathematically and depend on a large number of different factors, the task needs to be solved through the creation of a simulator.

This paper is organised as follows. We analyse IETF experience for SIP-signaling overload control problem solutions. Then we investigate overload control techniques which are implemented on the server and the client side. And finally, we introduce the architecture of the SIP simulation tool for modeling different overload control techniques.

Proceedings 27th European Conference on Modelling and Simulation ©ECMS Webjørn Rekdalsbakken, Robin T. Bye, Houxiang Zhang (Editors)
ISBN: 978-0-9564944-6-7 / ISBN: 978-0-9564944-7-4 (CD)

SIP OVERLOAD CONTROL PROBLEM OVERVIEW

Basic 503 mechanism

The SIP protocol provides a basic overload control mechanism through the 503 (Service Unavailable) response code. SIP servers that are unable to forward a request due to temporary overload can reject the request with a 503 response. The overloaded server can insert a Retry-After header into the 503 response, which defines the number of seconds during which this server is not available for receiving any further requests from the upstream neighbor. A server that receives a 503 response from a downstream neighbor stops forwarding requests to this neighbor for the specified amount of time and starts again after this time is over. Without a Retry-After header, a 503 response only affects the current request and all other requests can still be forwarded to this downstream neighbor. A server that has received a 503 response can try to resend the request to an alternate server, if one is available. A server does not forward 503 responses toward the UA and converts them to 500 Server Internal Error responses instead. The two possible scenarios of the mechanism 503 are shown in Fig. 1. Note that RFC 3261 provides, in the case of overload the recipient to discard incoming messages without notifying the sender.

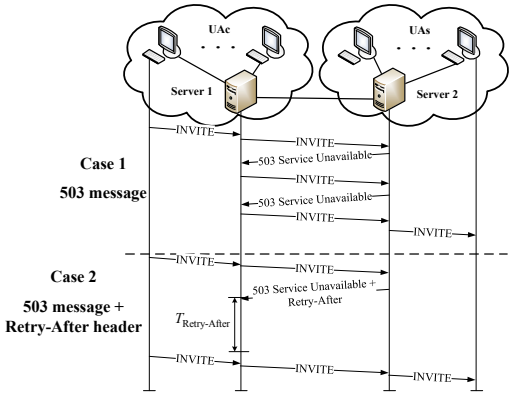

Figure 1: Two scenarios of 503 mechanism

A SIP server overload occurs if a SIP server does not have sufficient resources to process all incoming SIP messages. Several reasons, including Poor Capacity Planning, Component Failures, Avalanche Restart or Flash Crowds and the list of the problems that arise as a result of the 503 mechanism are presented in (RFC 5390, 2008).

Load Amplification. The supplementary result of the 503 mechanism is the tendency to amplify the load during periods of overload significantly, thus causing further aggravation of the problem and bringing the collapse of the network closer.

Underutilization. RFC 3261 does not cover how the 503 message recipient should react. In fact, there are some network configurations where it is not possible to clearly identify the sender of the message if the sender knows only a domain name of the cluster of receivers. So the sender may slow down the load to the entire cluster of servers with no overloaded servers, but not to the actual server in overload.

The Off/On Retry-After Problem. When the sender is balancing requests between a small number of the receivers, the 503 mechanism with Retry-After becomes noneffective because of its all-or-nothing technique. The 503 mechanism with Retry-After tends to cause highly oscillatory behavior under even mild overload.

Ambiguous Usages. The Standards do not clearly determine when the server must send the message with the code 503, and as a result of various implementations the message 503 is used to indicate different states. For example, according to (RFC 3398, 2002) the signaling gateway sends a message 503 in response to reports of inability to handle the request, which does not necessarily mean that the gateway is overloaded.

Rosenberg formulated 23 requirements to overload control mechanisms in (RFC 5390, 2008); mechanisms matching them will be able to predict and to avoid or quickly to cope with an overload on the server.

Explicit Overload Control Scheme

The problem domain of SIP overload control can be split into overload control between a user agent and a SIP server and overload control between SIP servers. The first document (RFC 6357, 2011) developed by SOC, contains the overload control mechanism classification with local, hop-by-hop and end-to-end schemes, and the following network topologies for "server-server" interoperation – load balancer, multiple sources and mesh.

Current work of the group is focused on the development of two hop-by-hop overload control schemes – Loss-based overload control and Rate-based overload control. The choice in favour of hop-by-hop mechanism was made because of the advantages over the other two mechanisms: the solutions implemented hop-by-hop scheme have better scalability and the scheme requires a SIP entity to aggregate overload status values of SIP servers only that each server communicates with.

The basic idea of LBOC scheme is that the sending entity (SE) reduces the number of messages on RE's request which will be send to the receiving entity (RE) by specified in the request amount of the total number of messages. RBOC scheme operates in the following way: RE informs SE about the maximum message rate which RE would like to receive from SE within a specified period of time. RE sends the control information to SE periodically depending on RE load changes.

Both of these schemes based on the idea of feedback control loop shown in Fig. 2 between all neighboring SIP servers that directly exchange traffic. Each loop controls only two entities. The Actuator is located on the sending entity and throttles the traffic if necessary. The receiving entity has the Monitor which measures the current server load.

The four Via header parameters ('oc', 'oc-algo', 'oc-validity' and 'oc-seq') are introduced in (IETF draft SIP

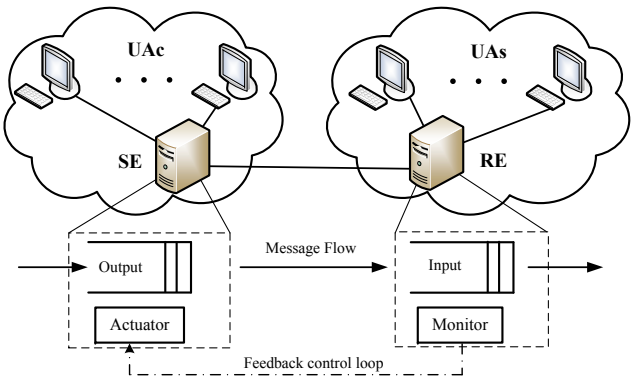

Figure 2: Feedback control loop between receiving and sending entities

Overload Control, 2013) to transfer the control information between two adjacent entities.

The integer parameter 'oc' consisting of 10 digits and its value defines what percentage of the total number of SIP requests are subject to reduction at the SE when the loss-based scheme is used. Analogously, when the rate-based scheme is used it indicates that the client should send SIP requests at a rate of 'oc'-value SIP requests or fewer per second. 'oc-algo' parameter defines the scope of algorithms supported by SE, e.g. oc-algo="loss", "rate". 'oc-validity' parameter contains a value that indicates an interval of time (measured in milliseconds) that the load reduction specified in the value of the 'oc' parameter should be in effect, its default value is 500 ms. 'oc-sequence' is the sequence number associated with the 'oc' parameter, timestamps usually use as its value.

The message format with the control information, the procedures of choice overload control algorithm, and the behavior of client and server using overload control mechanism are described in detail in (IETF draft SIP Overload Control, 2013; IETF draft SIP Rate Overload Control, 2013). However, these documents remains open the following questions that need further research:

- Criteria determining the choice of moments for sending messages with control information from SE to RE;
- Rule for choosing the value of 'oc' parameter;
- Rule for choosing the value of 'oc-validity' parameter.

To address these problems we have developed analytical models which we discuss in the next section.

OVERLOAD CONTROL TECHNIQUES OVERVIEW

Threshold Overload Control on the Server Side

As a criteria determining the choice of moments for sending messages with control information from SE to RE we propose to use hysteretic control technique. The system during operation changes its state depending on the total number of messages n present in it. Choose arbitrary numbers L and H such that $0 < L < H < B$, where B is the buffer capacity. When the system starts to work it is empty, $(n = 0)$, and as long as the total number of messages in the system remains below $H - 1$, system is considered to be in normal state, $(s = 0)$. When total number of messages exceeds $H - 1$ for the first time, the system changes its state to overload, $(s = 1)$, and RE informs SE that traffic load should be reduced: it stays in it as long as the number of messages remains between L and $B - 1$. Being in overload state, RE's system waits till the number of messages drops down below L after which it changes its state back to normal and informs SE about changes, or exceeds $B - 1$ after which it changes its state to blocking, $(s = 2)$, and ask SE for temporary suspension of sending SIP requests. When the total number of messages drops down below $H + 1$, system's state changes back to overload, and RE informs SE that the process of sending of messages can be resumed with the current limitations. Input load function $\lambda(s, n)$ is schematically depicted in Fig. 3.

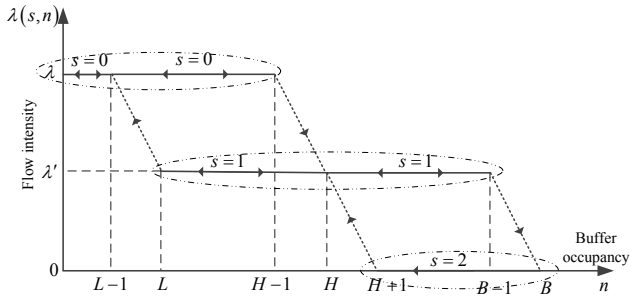

Figure 3: Hysteretic load control with two thresholds

Different analytical models in the form of a queuing system were built to estimate the performance indicators of hysteretic overload control algorithm (Abaev and etc., 2012, 2013). To estimate the optimal value of the thresholds, L and H, two optimization problems were formulated and solved in (Abaev et al., SOC simulation; Queueing System with Constant Service Time, 2012). In addition, such a system has been studied in the nonstationary case. The initial values of the thresholds were taken as a solution of the optimization problem from (Abaev et al., Analytical model for optimal SOC), i.e. $L = 74$ and $H = 85$. The dependency of the buffer occupancy on time t is shown in Fig. 4. The left Y-axis indicates the current buffer occupancy, $n(t)$, and the right Y-axis indicates the overload status of the system, $s(t)$. Server overload status changes are clearly shown in the two subfigures below the main one. Each subfigure depicts the chart to consistently zoom in on the previous figure.

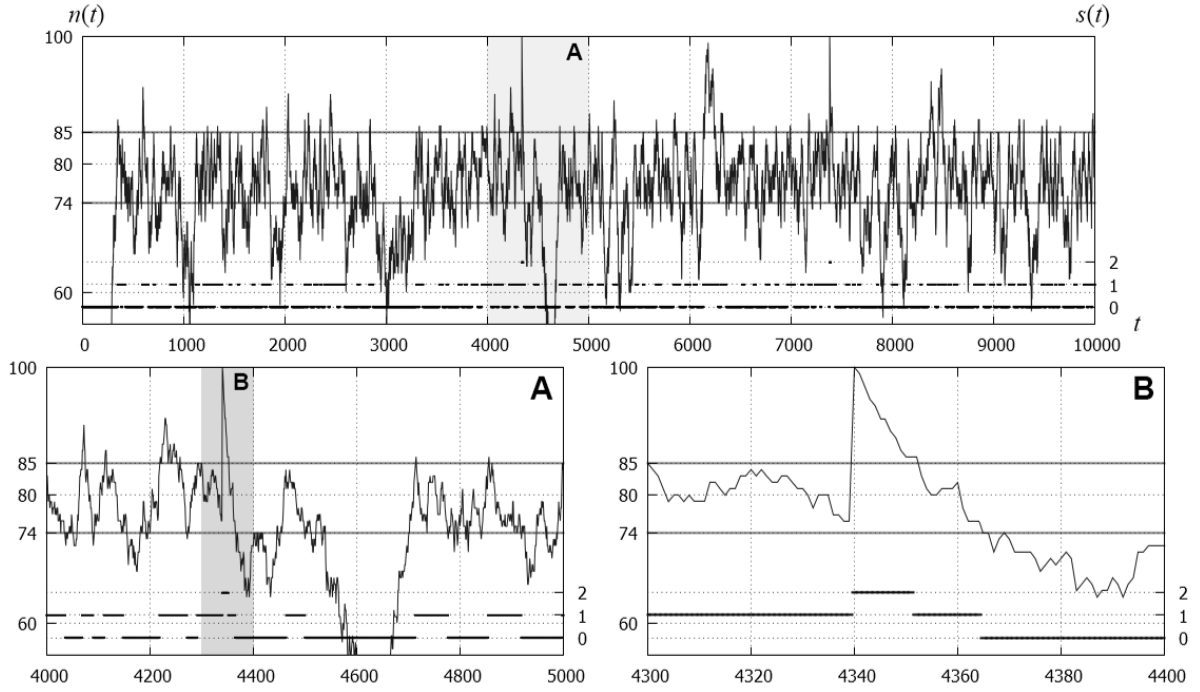

Figure 4: Modeling hysteretic overload control in unsteady condition

Default Algorithm on the Client Side for LBOC case

In the case of LBOC scheme the default algorithm for throttling incoming to the server traffic is used on the client side. The idea of the algorithm presented in (IETF draft SIP Overload Control, 2013) is to sift the client's outgoing flow. Let us consider the example of the implementation of the algorithm.

The client maintains two types of requests – the priority and non-priority. Prioritization of messages is done in accordance with local policies applicable to each SIP-server. In situations where the client has to sift the outgoing flow, it first reduces non-priority messages, and then if the buffer contains only priority messages and further reduction is still needed, the client reduces the priority messages.

Under overload condition, the client converts the value of the 'oc'=q parameter to a value that it applies to non-priority requests. Let N_1 denote the number of priority messages and N_2 denote the number of the non-priority messages in the client's buffer. The client should reduce the non-priority messages with probability $q_2 = \min\left\{1, q\frac{N_1+N_2}{N_2}\right\}$ and the priority messages with probability $q_1 = \frac{q(N_1+N_2)-q_2N_2}{N_1}$ if necessary to get an overall reduction of the 'oc' value.

To affect the reduction rate with probability q_2 from the non-priority messages, the client draws a random number between 1 and 100 for the request picked from the first category. If the random number is less than or equal to converted value of the "oc" parameter, the request is not forwarded; otherwise the request is forwarded. Recalculation of probabilities is performed periodically every 5-10 seconds by getting the value of the counters N_1 and N_2.

Leaky Bucket Algorithm on the Client Side for RBOC case

In the case of RBOC scheme the default Leaky Bucket algorithm (ITU-T I.371, 2004) is proposed to use on the client side to deliver SIP requests at a rate specified in the 'oc' value with tolerance parameter τ (IETF draft SIP Rate Overload Control, 2013). The Leaky Bucket algorithm can be viewed as a finite capacity bucket whose real-valued content drains out at a continuous rate of 1 unit of content per time unit and whose content increases by the increment T for each forwarded SIP request. T is computed as the inverse of the rate specified in the 'oc' value, namely $T = 1/oc$.

It is assumed that the client tries to put some content into the bucket at random times (time of a message arrival).

If at that moment the bucket capacity does not exceed the value τ, the incoming content (incoming message) is added to the bucket and increases the volume of content in the bucket by T. If at that moment the amount of content of the bucket more than τ, the incoming content (message) is not added to the bucket.

The random process of changes of the bucket's content is shown in Fig. 5. Let $X(t) \geq 0$ is the value of the bucket's content at the moment $t \geq 0$, t_k is the moments of arrival of the $k-th$, $k \geq 1$, message, $LCT(t)$ is the last confirmed time before moment t, $n(t)$ is the number of the latest message accepted before the moment t.

These variables satisfy the following relations:

$$n(t) = \sup_{k \geq 1} \{k : \ t_k < t, \ \ X(t_k) \leq \tau\}, \qquad (1)$$

$$LCT(t) = t_{n(t)}, \qquad (2)$$

$$X(t) = X(LCT(t)) - \ [t - LCT(t)]. \qquad (3)$$

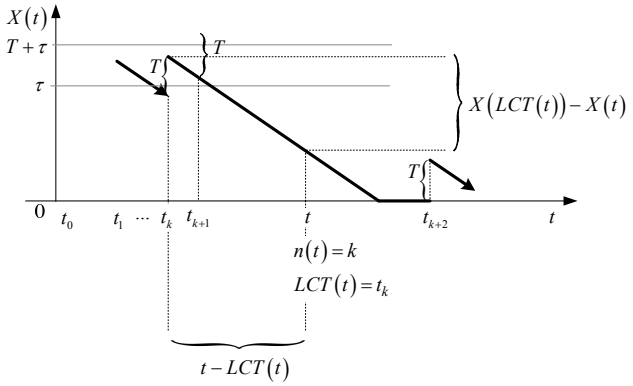

Figure 5: Changes of the value of the bucket's content $X(t)$

Note that the value τ is configured on the client side and chosen depending on the amount of the client's memory and its' processor performance. The problem of finding the optimal values of the parameters 'oc' will not be considered in this article because of space limitations.

OSN FUNCTIONAL REQUIREMENTS

Now we present the capabilities of simulation tool OSN (Open SIP signalling Node) which is under development and supports *nix like OS. Node's software architecture can be logically divided into three levels as shown in Fig. 6: TCP/IP Level, SIP Stack Level and Application Level.

TCP/IP Level corresponds to the first four levels of the ISO/OSI model and is implemented in OS, e.g. Debian or CentOS.

SIP Stack Level can be logically divided into two sublevels: the first one is SIP protocol sublevel which consists of SIP Manager, Message Processing, and Management Entity, the second one is Traffic Load Control sublevel which consists of Actuator, Monitor and Control Function.

The SIP protocol sublevel implements SIP baseline specification (RFC 3261, 2002; RFC 3263, 2002). New modules can be added if necessary. SIP Manager implements the client and server transaction state machines. Message Processing Module provides the complete set of tools for processing SIP messages. Management Entity module is responsible for dumping and aggregation of statistics. The Traffic Load Control sublevel is detailed described in the next subsection.

Application Level contains Presence and Location Modules. The Presence module allows a party to know the ability and willingness of other parties to participate in a call even before an attempt has been made. The module is responsible for handling Presence SUBSCRIBE requests with event package "presence" from Watchers, and enables the application to notify them about the Presence status of the Presentities.

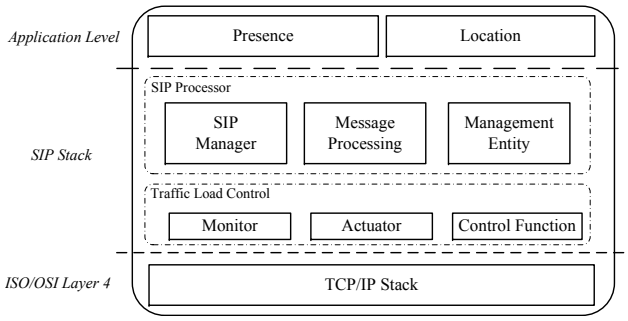

Figure 6: Architecture of the OSN simulation tool

Functional Requirements for Traffic Load Control sublevel

Each module of Traffic Load Control sublevel implements the functionality according to (RFC 6357, 2011). The communication between the modules is shown in Fig. 7. The Monitor measures the current load of the SIP Processor on the receiving entity. It implements the mechanisms needed to determine the current usage of resources relevant for the SIP Processor and reports load samples to the Control Function. The Control Function implements the overload control algorithm. The Control Function uses the load samples and determines if overload has occurred and a throttle needs to be set to adjust the load sent to the SIP Processor on the receiving entity. The Control Function on the receiving entity sends load feedback to the sending entity. The Actuator implements the algorithms needed to act on the throttles and ensures that the amount of traffic forwarded to the receiving entity meets the criteria of the throttle. The Actuator implements the algorithms to achieve this objective, e.g., using message gapping. It also implements algorithms to select the messages that will be affected and determine whether they are rejected or redirected.

In the case of LBOC and RBOC schemes Monitor watches the server's buffer occupancy and Control Function implements hystertic overload control. Module Actuator implements the default algorithm which was described above.

SUMMARY AND FURTHER STUDY

In this paper we give an overview of the problems and requirement analysis for SIP overload control mechanisms. We develop functional requirements for SIP simulation tool which are the result of the authors experience gained from the analytical modeling of overload control techniques performed by the authors in (Abaev and etc, 2012,

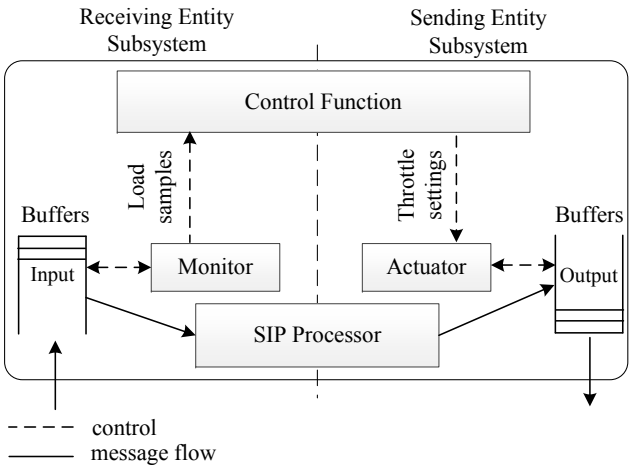

Figure 7: Communications between traffic control modules

2013). Future work will be focused on the development technical requirements of the simulator and mathematical modeling of overload control algorithms to make a proposal for the preliminary values of the control parameters, which will be used as input data for the simulator.

Acknowledgement This work was supported in part by the Russian Foundation for Basic Research (grants 12-07-00108 and 13-07-00665).

The author thank PhD student Andrey Samuylov for modelling hysteretic overload control in unsteady condition and magister Anastasia Khachko for figures design.

REFERENCES

Abaev, P., Gaidamaka, Yu., Samouylov, K. 2012. Modeling of Hysteretic Signaling Load Control in Next Generation Networks. Lecture Notes in Computer Science. Germany, Heidelberg, Springer-Verlag. –Vol. 7469. –P.371–378.

Abaev, P., Gaidamaka, Yu., Samouylov, K. 2012. Queuing Model for Loss-Based Overload Control in a SIP Server Using a Hysteretic Technique. Lecture Notes in Computer Science. Germany, Heidelberg, Springer-Verlag. -Vol. 7469. –P.440–452.

Abaev, P., Gaidamaka, Yu., Pechinkin, V., Razumchik, R., Shorgin, S. 2012. Simulation of overload control in SIP server networks. Proceedings of the 26th European Conference on Modelling and Simulation, ECMS 2012. –Germany, Koblenz. –Pp. 533–539.

Abaev, P., Pechinkin, V., Razumchik, R.. 2012. Analysis of Queueing System with Constant Service Time for SIP Server Hop-by-Hop Overload Control. Proceedings of the 4th International Congress on Ultra Modern Telecommunications and Control Systems, ICUMT 2012. –Saint–Petersburg, Russia. –Pp. 299–304.

Abaev, P., Pechinkin, V., Razumchik, R. 2012. On analytical model for optimal SIP server hop-by-hop overload control. Communications in Computer and Information Science: Modern Probabilistic Methods for Analysis of Telecommunication Networks. Germany, Heidelberg, Springer–Verlag. –Vol. 356. –P.1–10.

Abaev, P., Pechinkin, V., Razumchik, R. 2012. On Mean Return Time in Queueing System with Constant Service Time and Bi-level Hysteric Policy. Communications in Computer and Information Science: Modern Probabilistic Methods for Analysis of Telecommunication Networks. Germany, Heidelberg, Springer–Verlag. –Vol. 356. –P.11–19.

Rosenberg, J., Schulzrinne, H., Camarillo, G. et al. 2002. SIP: Session Initiation Protocol. RFC 3261.

Gurbani, V., Hilt, V., Schulzrinne, H. 2013. Session Initiation Protocol (SIP) Overload Control. draft-ietf-soc-overload-control-12.

Hilt, V., Noel, E., Shen, C., Abdelal, A. 2011. Design Considerations for Session Initiation Protocol (SIP) Overload Control. RFC 6357.

Noel, E., Williams, P. 2012. Session Initiation Protocol (SIP) Rate Control. draft-ietf-soc-overload-rate-control-03.

Rosenberg, J. 2008. Requirements for Management of Overload in the Session Initiation Protocol. RFC 5390.

Traffic control and congestion control in B-ISDN. ITU-T Recommendation I.371. 2004.

Camarillo, G., Roach, A., Peterson, J., Ong, L. 2002. Integrated Services Digital Network (ISDN) User Part (ISUP) to Session Initiation Protocol (SIP) Mapping. RFC 3398.

Rosenberg, J., Schulzrinne, H. 2002. Session Initiation Protocol (SIP): Locating SIP Servers. RFC 3263.

AUTHOR BIOGRAPHIES

PAVEL O. ABAEV received his Ph.D. in Computer Science from the Peoples' Friendship University of Russia in 2012. He has been a senior lecturer in the Telecommunication Systems department of the Peoples' Friendship University of Russia since 2011. His current research focus is on NGN signalling, QoS analysis of SIP, and mathematical modeling of communication networks. His email address is `pabaev@sci.pfu.edu.ru`.

YULIYA V. GAIDAMAKA received the Ph.D. in Mathematics from the Peoples' Friendship University of Russia in 2001. Since then, she has been an associate professor in the university's Telecommunication Systems department. She is the author of more than 50 scientific and conference papers. Her research interests include SIP signalling, multiservice and P2P networks performance analysis, and OFDMA based networks. Her email address is `ygaidamaka@sci.pfu.edu.ru`.

KONSTANTIN E. SAMOUYLOV received his Ph.D. from the Moscow State University and a Doctor of

Sciences degree from the Moscow Technical University of Communications and Informatics. During 1985–1996 he held several positions at the Faculty of Sciences of the Peoples' Friendship University of Russia where he became a head of the Telecommunication Systems Department in 1996. His current research interests are performance analysis of 3G networks, teletraffic of triple play networks, and signaling networks planning. He is the author of more than 100 scientific and technical papers and three books. His email address is ksam@sci.pfu.edu.ru.

SERGEY YA. SHORGIN received a Doctor of Sciences degree in Physics and Mathematics in 1997. Since 1999, he is a Deputy Director of the Institute of Informatics Problems, Russian Academy of Sciences, since 2003 he is a professor. He is the author of more than 100 scientific and conference papers and coauthor of three monographs. His research interests include probability theory, modeling complex systems, actuarial and financial mathematics. His email address is sshorgin@ipiran.ru.