Analyze of SIP Messages and Proposal of SIP Routing

F. Csoka, I. Baronak, E. Chromy and L. Kockovic

***Abstract*—**This paper deals with the functionality of SIP and design of an efficient and optimized process for routing SIP messages. It is used for creation of VoIP calls. This routing logic, in a form of a script, should be faster and simpler than current implementations. It should not include any functionality that is not necessary for initiating VoIP calls on LAN.

***Keywords*—**efficient routing, Kamailio, SIP.

I. Introduction

NOWADAYS there is a large number of multimedia applications, which require a creation and a management of multimedia session for their correct operation. In most cases, the session consists of constant exchange of data between two or more end users.

We must take into consideration the fact that this communication is made difficult by the abilities of the end users. They can connect to a network and move freely within it. Their connection to the network may change with the change of an end point they are currently connected to. The end users may also address each other with different names. There is a multitude of protocols that can transfer media such as voice, image, text and data in real time [11].

The Session Initiation Protocol (SIP) [1], [2] cooperates with them. It enables the end users, known as User Agents (UA), to find one another on the network and negotiate the parameters of session. SIP allows the creation of network infrastructure [5], [7], [9] consisting of UA and SIP servers [3], which process and route request from users.

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 Program for the project University Science Park of STU Bratislava, ITMS 26240220084, co-funded by the European Regional Development Fund. This article is a part of research activities conducted at Slovak University of Technology Bratislava, Faculty of Electrical Engineering and Information Technology, Institute of Telecommunications, within the scope of the project "Grant programme to support young researchers of STU - Optimization of resources allocation in IPTV systems".

F. Csoka is with the Institute of Telecommunications, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovakia; e-mail: filip.csoka@gmail.com.

I. Baronak is with the Institute of Telecommunications, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovakia; e-mail: baronak@ut.fei.stuba.sk.

E. Chromy is with the Institute of Telecommunications, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovakia; e-mail: chromy@ut.fei.stuba.sk.

L. Kockovic is with the Institute of Telecommunications, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovakia; e-mail: kockovic@ut.fei.stuba.sk.

II. Current State

The vast majority of existing open source SIP servers are too complex to be used for testing SIP User Agent applications. Even those designed to run on LAN contain abundance of function, which are counterproductive from phone application (tested point of view). They have been designed to offer as much functionality as possible in reasonable amount of time, with very little optimization.

III. SIP

SIP is an application layer protocol that creates, modifies and terminates multimedia sessions [10], [12]. It is most commonly used to make VoIP calls, although it can have other applications as well. For example, it can be used to initiate direct file transfers between end users. SIP protocol does not provide any service, rather it allows implementation of various services.

It operates on a simple REQUEST → RESPONSE principle. Clients generate requests and receive responds from other clients or servers. The syntax of SIP is fairly similar to HTTP. A specific form of URI (Uniform Resource Identifier) called "SIP URI" is used for addressing.

There are six request messages defined in RFC 3261 and all have a fairly similar structure. They consist of Request-Line followed by several header fields. The Request-Line comprises of the used request method (for example INVITE, REGISTER, BYE …) and the SIP URI of the recipient of the request. Following header fields have all the same structure: "name of the field: value assigned to that field". For example the header field max-forwards may look like this: "Max-Forwards: 70". The mandatory header fields of INVITE request (the message used to initialize connection) are:

1) Via – fields are added to the original message by SIP servers routing the message, they contain addresses and information used for routing of responds,
2) From – SIP URI of the user that send the request,
3) To – SIP URI of a user to whom the request is addressed,
4) Contact – contact field contains an IP address that can be used to contact the sender of SIP message directly,
5) Call-ID – should contain identifier of call that is unique across the network,
6) CSeq – consists of a number that is incremented during

call with every new generated request,

7) Max-Forwards – is an equivalent of time-to-leave field from IP, it is decremented each time the message is processed by a SIP server.

Additional header fields may be present, depending on type of request and the service requested by the message.

A SIP message may consist of a body following the header fields. Additional information may be encoded in it using SDP or XML (such as supported codecs, status of the device …).

SIP response messages have almost the same structure as requests, except the Request-Line is replaced by Status-Line. It describes the response to our request using a code (codes almost the same as in HTTP) and a reason phrase that explains what the code means to a human reading it, for example "SIP/2.0 200 OK" response message means that our request has been accepted by its recipient. The header fields used in responds are the same one as those used in requests.

A. The Algorithm Proposal

Our goal was to design a highly efficient and optimized algorithm that analyzes and routs SIP messages. This algorithm should only take actions intended to route the message as quickly as possible. Without adding any unnecessary delay to this process. To achieve that, we needed to design an algorithm, which would allow devices to register and route messages in the most efficient possible manner.

The easiest way to achieve this, was to create a simple software that constantly listens at chosen port (standard SIP port 5060). And analyse all incoming SIP messages, saves them into pre-allocated structures and variable, thus making individual header fields of the message easily addressable.

The next step was to create a routing script. A short program that is executed for each of these received SIP messages regardless of their type. The function of our program is depicted by following flow diagrams.

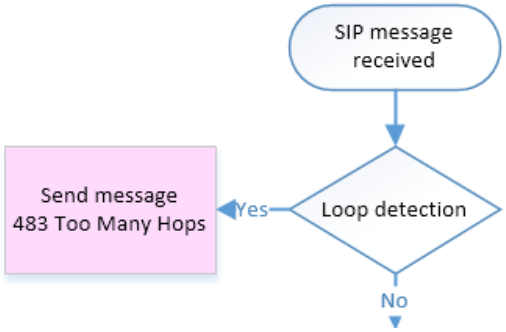

Fig. 1 loop detection

First step (Fig. 1) in our message processing is checking the variable corresponding to contents of Max-Forwards header field. If the value is zero, it is safe to assume that the message is either looped, or it cannot reach its recipient. We must generate a SIP response message "483 Too Many Hops" with header fields corresponding to the original request and send it to the source of the request. If the value is not zero, we decrement it by one and continue with message processing.

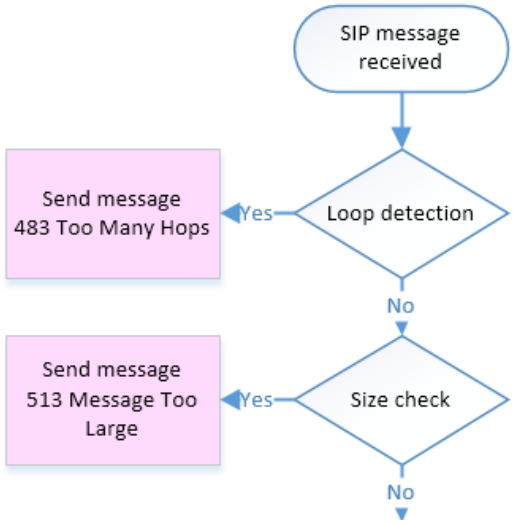

Fig. 2 size check

Next step is size check (Fig. 2). We cannot allow users to generate requests of unlimited size, it could overload our system. There is no limit to what can be attached to the body of a SIP message. A picture to identify the caller, compressed video and many others. That is why we have to discard messages with size larger than reasonable number. We chose two megabytes as limit value. Any message larger than 2 MB will be discarded and its sender is send SIP response message "513 Message Too Large". If the size is smaller, we can continue with message processing.

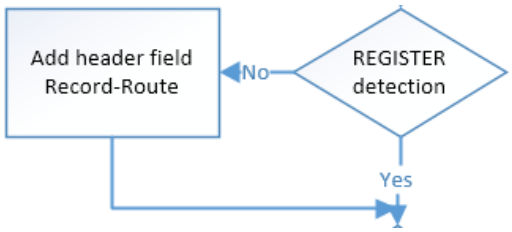

Fig. 3 header insertion

These first two steps of our script were only preliminary checks and are followed by a short informative block. Since we have chosen that our server should work as a SIP Proxy server, we must be able to inform the end users about this fact. The simplest way to do that, is to insert a new header field Record-Route (Fig. 3) to every message before we forward it to its destination. Obviously, it is not necessary to add anything to REGISTER message. It is addressed to server itself and is not forwarded further.

Presence of Record-Route header fields informs end points (for example SIP phones), that the routing device should be addressed all signaling communication, until the end of call (or session). Depending on their configuration the end devices may choose to ignore this information. It could cause the

server performance issues when handling large amount of calls. However this problem will surface when this routing algorithm is not programmed directly, but implemented in SIP server that monitors transaction (such as Asterisk, Kamailio …) [4], [6], [8].

After the preliminary inspection of the received SIP message is concluded and the informative header field is inserted we may begin the routing of the message. Statistically fifty percent of all processed messages are requests and fifty percent are responds (during basic call setup server routes 6 messages, 3 request and 3 responds). Responds carry within them addresses of the next hop and entire routing path. That is why it is extremely important to check whether the message is a response before we decide to make any time consuming operation (such as searching through database).

Fig. 4 response detection

The server must digest (Fig. 4) his own Via header field. It is deleted from the message and the address in the next Via field is used as a destination address for adjusted message. The message is then recreated from the variables and structures, which were filled by parsing the original message and then adjusted by the routing process so far. Message generated this way is subsequently sent to the IP address in the top Via header field. Approximately fifty percent of messages are processed this way.

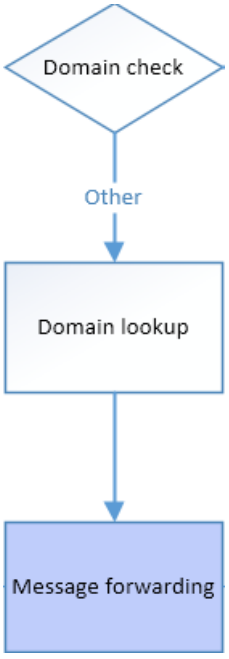

Fig. 5 domain check

The remaining fifty percent is harder to process from the point of required time. They are requests and their destination needs to be found.

First step in this process is domain check (Fig. 5). SIP server checks whether the message is addressed to him or to a device registered in the domain of our server. If the message is addressed to a different domain from our own, the server finds the address of this domain in his database and adds new Via header field with his information to the received message. The message is reconstructed same as before and sent to the address found during this domain lookup. If the name of the address is not found in the database, the message is sent to the first domain on the list. This gives us at least some chance that the message gets delivered. Since it is unlikely that other servers will register at our server, the domains and their addresses need to be edited to the database manually.

If the message is addressed to our domain, we must consider two possibilities. First is that the message is addressed to the server itself. In our server architecture, it can only be addressed to the server in registration. So the routing process will continue by checking whether the message is a registration.

Fig. 6 REGISTER detection

In case of positive REGISTER detection (Fig. 6), the server has to make a new AOR (address of record) database entry, for which the user name in SIP URI is extracted along with the contact address in the Contact header field. Once the AOR is created server can discard the original request (since its purpose has been fulfilled). And inform sender of successful registration in a form of "200 OK" SIP message, which is one again generated from the header fields of original request.

The process if fairly similar when the received request is not addressed to the server, but rather to a user registered at our server.

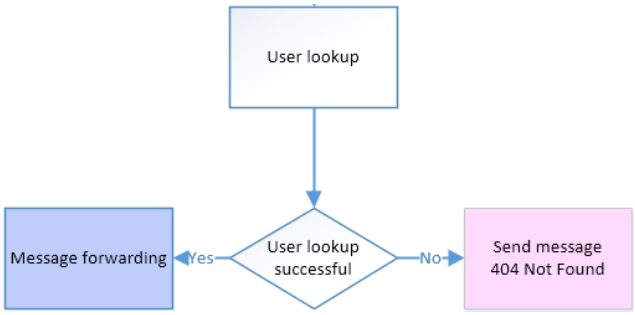

Fig. 7 user lookup

Please notice that the most time and resources consuming process, which is user lookup (Fig. 7), has been saved for last.

It may take considerably more time than the rest of the routing combined. Based on the size of the database and number of registered users. If the user with corresponding name is not found, server generates the well-known message "404 Not Found" and sends it to the sender of the request. The message is once again generated using the information contained in the original request. The message may be discarded. If the lookup was successful, the request gets reconstructed and a new Via header field is attached to it. The message is then forwarded to the address found in the database.

There are only three ways in which this routing algorithm may end (Fig. 8). First is an error state, during which the original request is discarded and sender is informed why the request was invalid. These states are depicted red. Second is acceptance, the algorithm has received valid registration in form of REGISTER message and accepted it. The sender is informed by a "200 OK" message. This state is shown as green. The last state is forwarding of the received message to its desired destination. This state is Blue.

IV. Conclusion

Although this script may seem complicated, it is actually a bare minimum needed for ensuring functionality of Session Initiation Protocol in computer networks. It has been optimized and any reduction of processing steps would result at least in possible errors, but most likely in collapse of the communication. It is possible to implement support for additional services and expand at the cost of efficiency and speed of routing. The testing had been executed by replacing routing script of Kamailio with our own. Basic LAN configuration provided with this system had an average call rate of 17962.381 calls per second. On the same hardware configuration and the same Kamailio server with our routing script, the system had a call rate of 38975.679 calls per second. This configuration is ideal for endpoint application testing, because it does not introduce any unnecessary delay into message routing process.

References

[1] *SIP: Session Initiation Protocol*, RFC 3261, 2002.

[2] A. Johnston, *SIP: Understanding the Session Initiation Protocol*. Norwood: Artech House, 2009, 395 pages.

[3] *Session Initiation Protocol (SIP): Locating SIP Servers*, RFC 3263, 2002.

[4] Kamailio SIP Server Wiki. (2015, October 23). Kamailio SIP Server (SER): New Features in v4.1.x [Online]. Available: http://www.kamailio.org/wiki/features/new-in-4.1.x

[5] F. Gonclaves, *Building Telephony Systems with OpenSIPS 1.6*. Birmingham, UK, 2010, 264 pages.

[6] B.-A. Lancu, D.-C.Mierla, (2015, October 23) *Kamailio (OpenSER) 1.2.0 - Performance Tests* [Online]. Available: http://www.kamailio.org/docs/openser-performance-tests/

[7] F. Gonclaves, *Building Telephony Systems with OpenSER*. Birmingham, UK, 2008, 303 pages.

[8] Kamailio SIP Server Wiki. (2015, October 23). Kamailio SIP Server v4.1.x (stable): Core Cookbook [Online]. Available: http://www.kamailio.org/wiki/cookbooks/4.1.x/core#loadmodule

[9] V. Kumar, M. Korpi, S. Sengodan. "IP Telephony with H.323 Architectures for Unified Networks and Integrated Services", John Wiley & Sons, 2001, 605 pages.

[10] M. Voznak, J. Rozhon, "Methodology for SIP infrastructure performance testing", *WSEAS Transactions on Computers*, vol 9, issue 9, 2010, pp. 1012-1021.

[11] M. Halas, S. Klucik, S, "Modelling the probability density function of IPTV traffic packet delay variation", *Advances in Electrical and Electronic Engineering*, vol. 10, issue 4, 2012, pp. 259-263, doi: 10.15598/aeee.v10i4.726.

[12] M. Voznak, J. Rozhon, "Approach to stress tests in SIP environment based on marginal analysis", *Telecommunication Systems*, vol. 52, issue 3, 2013, pp. 1583-1593, doi: 10.1007/s11235-011-9525-1.

F. Csoka was born in Bratislava, Slovakia in 1992. He is a student at the Institute of Telecommunications, Faculty of Electrical Engineering and Information Technology of Slovak University of Technology (FEI STU) Bratislava. He focuses on VoIP scalability and security.

I. Baronak was born in Zilina, Slovakia, on July 1955. He received the electronic engineering degree from Slovak Technical University Bratislava in 1980. Since 1981 he has been a lecturer at Department of Telecommunications STU Bratislava. Nowadays he works as a professor at Institute of Telecommunications of FEI STU Bratislava. Scientifically, professionally and pedagogically he focuses on problems of digital switching systems, ATM, Telecommunication management (TMN), NGN, IMS, VoIP, QoS, problem of optimal modeling of private telecommunication networks and services.

E. Chromy was born in Velky Krtis, Slovakia, in 1981. He received the Master degree in telecommunications in 2005 from Faculty of Electrical Engineering and Information Technology of Slovak University of Technology (FEI STU) Bratislava. In 2007 he submitted PhD work. Nowadays he works as assistant professor at the Institute of Telecommunications of FEI STU Bratislava.

L. Kockovic was born in Nove Zamky, Slovakia, in 1987. He received the Master degree in telecommunications in 2012 from Faculty of Electrical Engineering and Information Technology of Slovak University of Technology (FEI STU) Bratislava. In his PhD work he focuses on the IPTV and video delivering for users in wireless networks. Nowadays he works as assistant professor at the Institute of Telecommunications of FEI STU Bratislava

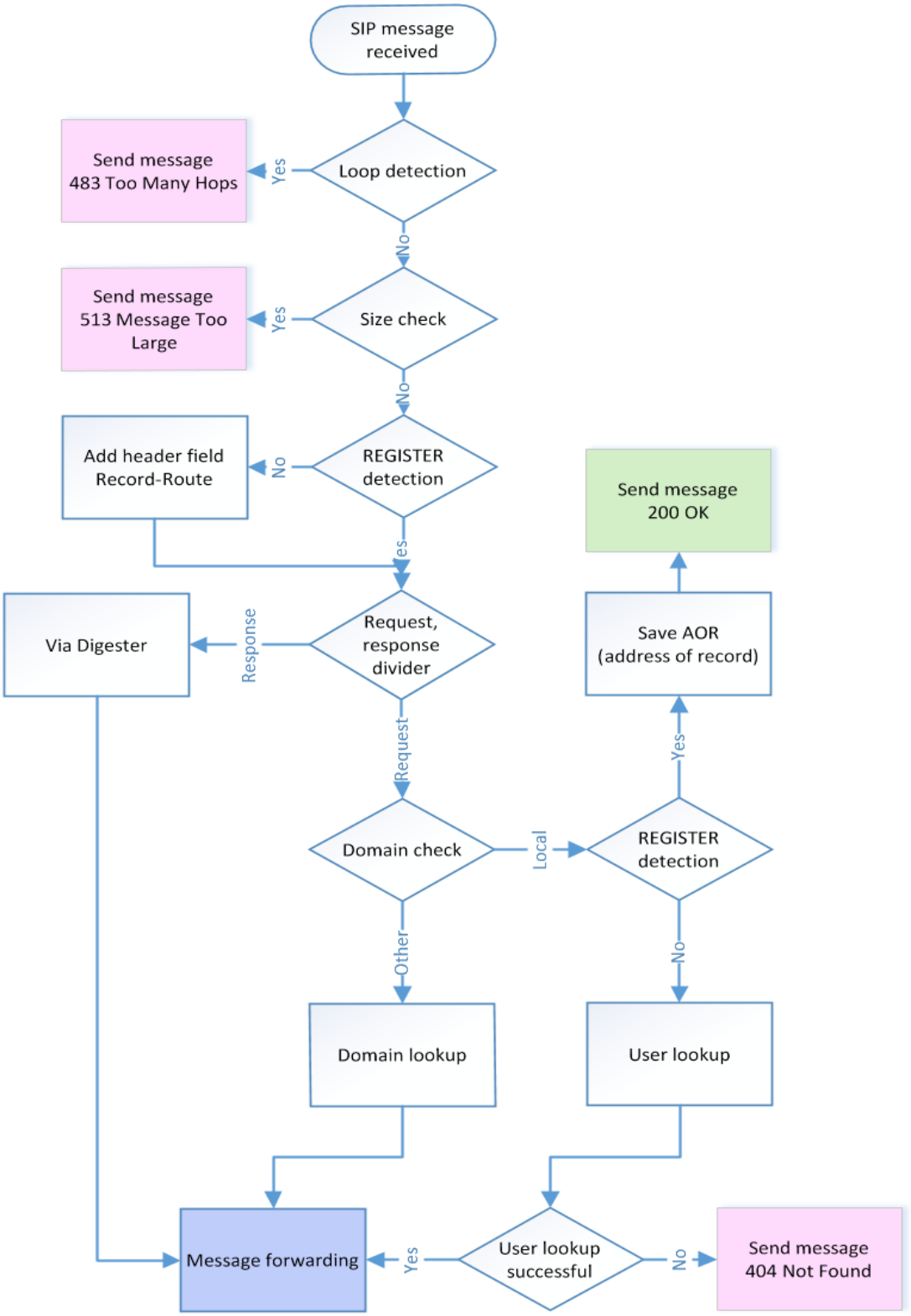

Fig. 8 routing logic