SIP PROTOCOL MODEL FOR OMNeT++

Jan KUCERAK, Petr CHLUMSKY

Department of Telecommunication Engineering, Faculty of Electrical Engineering,
Czech Technical University in Prague, Technicka 2, 166 27 Prague, Czech Republic

jan.kucerak@fel.cvut.cz, petr.chlumsky@fel.cvut.cz

DOI: 10.15598/aeee.v14i3.1502

Abstract. *The article describes our new SIP protocol implementation for the OMNeT++ simulation framework. OMNeT++ simulation framework provides an extensive support of IP related protocols, nevertheless a working SIP protocol implementation is missing. Real measurements were also done using a SIPp traffic generator and the results are compared to those obtained by our new SIP model. Since this work is a part of bigger project concerned strictly on measuring "first response times" over networks with a faulty transmission links, the actually collected statistics are focused only this way.*

Keywords

Network simulation, OMNeT++, SIP, VoIP.

1. Introduction

Working on the project that focuses on research of universal integrated IP (Internet Protocol) radio gateway for implementation into dispatching system we encountered a problem of simulating a VoIP (Voice over Internet Protocol) communication. For our network simulations we use simulation framework OMNeT++ [1], which proved to be very useful because of its well known capabilities on the field of telecommunication networks simulation. Although there are many extensive libraries of communication protocols and nodes for this simulation tool, currently there is no working SIP (Session Initiation Protocol) protocol implementation available for actual(4.x) version of OMNet++ tool. As some papers indicate [2] and [3], there are proprietary SIP protocol simulation tools available, such as OPNET Modeler [4]. Nevertheless, we prefer to use OMNeT++ simulation framework with the INET library, mostly because of its open sources and complete implementation of lower RM OSI layers. Moreover OMNeT++ is free of charge for academic purposes. Therefore we decided to write our own implementation of SIP protocol for OMNet++.

The rest of the paper is organized as follows. Firstly, we discuss proposed simulation model and its implementation in OMNeT++ simulation framework. OMNeT++ is introduced in the Subsection 2.1. . that is followed by the description of implementation. Section 3. presents data from the simulation and from the measurement and the comparison is shown in Section 4. . Finally the summary is presented in Conclusion with our idea of the future work.

2. Simulation Model

We used simulation framework OMNeT++ with library INET for our simulation of SIP signalling communication. On top of INET we implemented new modules that allowed a basic simulation of SIP protocol function. Therefore, we were able to simulate the most crucial functions of SIP, such as registration and connection establishment procedures. With respect to main project goal only a basic "INVITE-RING-OK-BYE" scenario was run. Simulation was run over and over with varying link delay and packet error rate parameters, measuring the time to the first response received by client. Of course any other desired types of statistics could be easily obtained by simulation model, but this was not a goal for now.

2.1. OMNeT++

OMNeT++ is a development framework for a simulation of discrete events based systems where a communication model is decomposable into messages. Simulations designed in OMNeT++ are scalable due to hierarchical modules division and because of object oriented approach it is possible to create complex network models in a relatively short time. Basic element

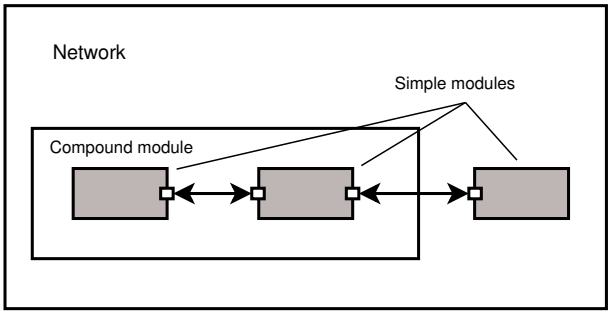

Fig. 1: Simple and compound modules.

of simulation model is so-called simple module Fig. 1. This simple module executes desired functions, it is the core of each more complex module. Simple modules are coded in C++ language. More complex modules are connected in hierarchical order by the NED (Network Description) language. These modules can be connected together and, therefore, create equivalent of network node (router, computer, access point). Furthermore, nodes connected together can create whole communication network.

Common network protocols, throughout the RM OSI layers (e.g. IEEE 802.11g, MPLS, IPv4, TCP, OSPF, DHCP), are implemented in modules library called INET [5]. Despite extensive number of modules there are still no modules available for SIP protocol simulation.

2.2. The SIP Protocol Model

SIP is a communication protocol for signaling and controlling multimedia communication sessions capable of running on different transport layers, e.g., TCP, UDP or SCTP. Our implementation of SIP protocol is based on the RFC 3261 [6]. We use UDP as a transport protocol given the scale of its common usage compared to other transport protocols. Our SIP protocol implementation is fully connected with INET library, both parts of the communication (SIP UA client and SIP proxy server) are designed as a UDP applications for the StandardHost compound module. These two applications handle their own communication over the common OSI protocol stack, that is already implemented in INET library. A simple DNS functionality was implemented as well, in order to enable true SIP-URI to IP address translations. This was done as a preparation for possible further model extension. Our actual DNS model works with ideal/zero delay for now. Of course in real world DNS queries delays can increase response time results, but this was not the actual scope.

Our design requires extension of the configuration file by these items:

- `address` automatically assigned address by the network address configurator module,
- `port` where the application will listen on,
- `timeToRegister` time between the start of simulation and the first try to register client to the proxy server,
- `timeToCall` time to schedule the INVITE message,
- `timeToResponse` time for simulation of response processing,
- `timeToAnswer` time of ringing, ends with call accept or deny,
- `timeCallSeason` duration of the call,
- `SIPnameClinet` the first part of SIP URI (Uniform Resource Identifier) that represents name of UA (User Agent),
- `SIPserver` the second part of SIP URI that represents name of proxy server,
- `connectAddress` SIP proxy server address,
- `SIPserverName` name of the SIP server.

Communication dialog as a set of variables is defined between the two parts of communication. This set is held only within the communication and is deleted at the end of call. Dialog consists of this variables:

- CallID,
- LocalTag,
- RemoteTag,
- LocalSeqnum,
- RemoteSeqnum,
- LocalUri,
- RemoteUri,
- RemoteTarget,
- Via.

The triplet CallID, RemoteTag and LocalTag is used for precise dialog identification. CallID is generated by the calling UA. Calling UA inserts its Tag into the INVITE message. The called part of communication saves this tag and generates its own tag into the response message. After this exchange the Tag (transaction number) is inserted in every following message. LocalSeqnum is number incremented by every sent request message. RemoteSeqnum is set during the request processing. Local and RemoteUri are addresses

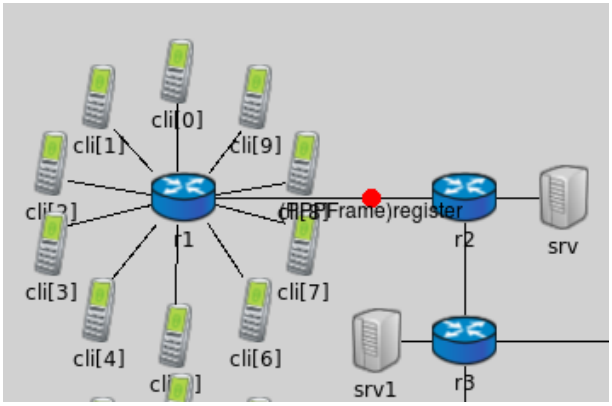

Fig. 2: Part of demonstration testing network (OMNeT++ GUI screenshot).

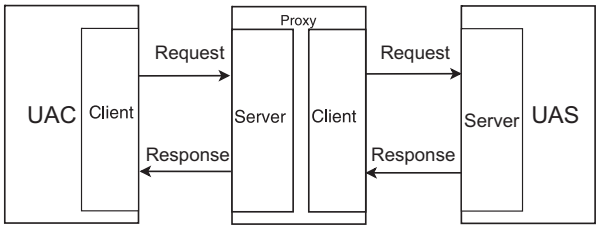

Fig. 3: Communication between SIP devices.

of appropriate parts of communication, RemoteTarget represents the remote target. Via shows the transport protocol used and the request route, each proxy adds a line to this field. The UDP transport timers from RFC 3261 are implemented in simulation model as well and can be altered arbitrarily.

Demonstration testing network used for our SIP protocol implementation testing is shown in Fig. 2. Only simple routers were used to simulate the IP network, but basically any network modules and topology available in INET framework can be simulated.

3. Real measurement

SIPp is a free open source command line based test tool and traffic generator for the SIP protocol. It includes a several basic SIPstone [7] user agent scenarios. More information about this tool can be found in papers [8] and [9], where authors introduced a new approach to SIP performance testing based on SIPp generator and complete list of SIPp features is available on the projet website [10]. Generic Linux machines were used as clients running standard SIPp UAC/UAS scenarios. Kamailio [11] software was used for SIP proxy servers. The network topology was pretty much the same as the one used in simulation. SIP proxies were connected to nearly "ideal" IP infrastructure (1Gb switches), while all the SIP clients were connected to the same infrastructure through a hardware network emulator Simena NE2000 [12] which provided a controlled network parameters downgrading. Rather overpowered machines were used in order to ensure that most of network performance degradation is really caused by the emulator.

Actual steps used for emulated latency were: 0 to 500 ms in 13 uniform steps(x axis of Fig. 4). Packet error rate simulated by NE2000 network emulator was 0; 0.1; 0.2 and 0.3 (y axis of Fig. 4). These bounds were chosen experimentally. The reason for this was, that all the measurements had to be concerned on "time to first reply" parameter investigation. It makes sense to measure such a parameter only if there is any reply at all. So we decided to narrow the measurement to the cases, where less than 3 of 250 calls attempted by client failed completely. The time to first server response was measured on clients side. 250 calls were done at each latency-error rate combination step and the average response time was counted (z axis Fig. 4 in seconds).

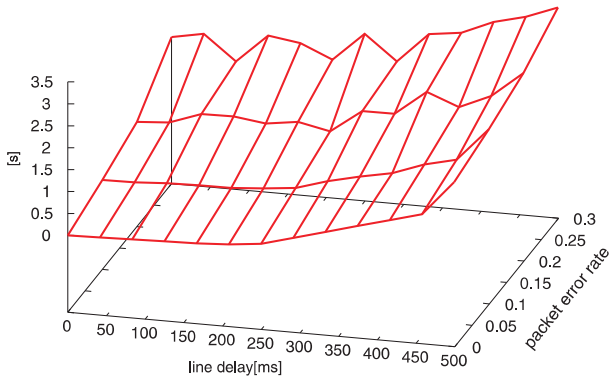

Fig. 4: SIPp measurement results.

4. OMNeT++ simulation

Similar settings as used in real measurement were set for OMNeT++ simulation. Network parameters degradation previously done by NE2000 was now simulated by clients link with pre-set packet delay and packet error rate. All the other links were set to ideal (zero loss & delay). Same 250 calls simulation was run for every loss-delay combination. The simulation time from a first INVITE being sent by client module to the first end to end reply being received was measured for every call and an average value was counted. Obtained results can be seen in a graph Fig. 5 with very same axis setout as the previous one (z axis represents computed average response times in seconds).

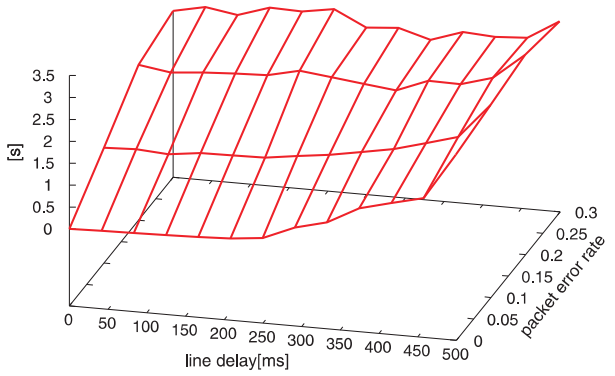

Fig. 5: OMNET++ simulation results.

5. Conclusion

Under higher error rates conditions OMNeT++ simulation results are obviously more uniform than those obtained by real measurement. We explain this by the fact that at the end OMNeT++ simulation is still an ideal environment where every faulty behaviour have to be set explicitly thus the results are predictively more steady.

Real measurement on the other side gives slightly better times in overall. Total average of differences is 0.36 seconds in favour of real measurement. Most of this is caused by slightly better results of real measurement in highest error rate conditions. We explain this by the fact, that the Kamailio SIP proxy software as a stateful SIP proxy tries bit harder to deliver the messages than our simulation SIP proxy module, for example by sending more copies of a retransmitted message once the original one was lost.

Anyway, both graphs clearly show very similar trends. Despite our actual OMNET++ SIP protocol model implementation is still incomplete it already shows a solid correlation with real SIP measurements at least when it comes to first response times parameter watching.

In the future work we intend to focus on extending the model to involve full transport implementation(TCP and SCTP). Also the connection between SIP signalization and VoIP audio streams can be implemented. Moreover, we plan to implement better quality of communication evaluation and system dimensioning for different network scenarios.

Acknowledgment

This work was supported by the Grant of the Technology Agency of the Czech Republic, No. TA04031186, „Universal Radio Gateway for IP Communication in Dispatching Systems“, and was researched in cooperation with TTC Telekomunikace.

References

[1] VARGA, A. and R. HORNIG. An overview of the omnet++ simulation environment. In: *1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops, ser. Simutools*. Brussels: ICST, 2008, pp. 1–10. ISBN 978-963-9799-20-2. DOI: 10.1.1.231.4511.

[2] RONG, B., J. LEBEAU, M. BENNANI, A. ELHAKEEM and M. KADOCH. OPNET simulation of SIP based SIP telephony over MPLS network. In: *OPNETWORK*. Washington: Opnet, 2004, pp. 1–7. ISBN 0-13-046099-0.

[3] ATRIANFAR, H. and Z. AYATOLLAHI. Expansion of opnet modeler sip model for performance evaluation of hierarchical call routing in NGN. In: *System Simulation and Scientific Computing*. Beijing: IEEE, 2008, pp. 179–185. ISBN 978-1-4244-1786-5. DOI: 10.1109/ASC-ICSC.2008.4675352.

[4] Riverbed: Steelcentral for performance management and control. *Riverbed* [online]. 2015. Available at: `http://www.riverbed.com/gb/products/steelcentral/opnet.html`.

[5] INET Framework: An open-source OMNeT++ model suite for wired, wireless and mobile networks. *INET Framework* [online]. 2016. Available at: `https://inet.omnetpp.org/`.

[6] RFC 3261. *Sip: Session initiation protocol*. New York: Internet Engineering Task Force, 2002.

[7] SCHULZRINNE, H. G.; S. NARAYANAN, J. LENNOX and M. DOYLE. SIPstone: Benchmarking SIP Server Performance. In: *Columbia Academic Commons* [online]. 2002. Available at: `http://academiccommons.columbia.edu/catalog/ac:109963`.

[8] VOZNAK, M. and J. ROZHON. Approach to stress tests in sip environment based on marginal analysis. *Telecommunication Systems*. 2013, vol. 52, no. 3, pp. 1583–1593. ISSN 1018-4864. DOI: 10.1007/s11235-011-9525-1.

[9] VOZNAK, M. and J. ROZHON. Methodology for SIP infrastructure performance testing. *WSEAS Transactions on Computers*. 2010, vol. 9, iss. 9, pp. 1012–1021. ISSN 1109-2750.

[10] SIPp: Reference documentation. *SIPp* [online]. 2014. Available at: `http://sipp.sourceforge.net/doc/reference.html`.

[11] The Open Source SIP Server. *KAMAILIO* [online]. 2016. Available at: https://www.kamailio.org/w/.

[12] Simena network emulator. *NETSCOUT* [online]. 2016. Available at: http://www.netscout.com/.

About Authors

Jan KUCERAK was born in Sokolov, Czech Republic. He received his M.Sc. from the Armed Forces Academy of gen. M. R. Stefanik in 2005. His research interests include telecommunication protocols and signalling.

Petr CHLUMSKY was born in Kladno, Czech Republic. He received his M.Sc. from the Czech Technical University in Prague in 2010. In 2015 he received the Ph.D. degree in telecommunication engineering from Czech Technical University in Prague, Faculty of Electrical Engineering. His research interests include wireless transmission, network coding and network simulation.