Telecommun Syst (2015) 59:541–549
DOI 10.1007/s11235-014-9914-3

Advanced solution of SIP communication server with a new approach to management

Miroslav Voznak · Jiri Slachta · Lukas Macura · Karel Tomala

Published online: 22 November 2014

Abstract The paper deals with the development of an advanced solution for a SIP communication server. Works were carried out within the scope of Bright Embedded Solution for IP Telephony (BESIP) project. The output of the project should be a modular architecture with additional functionality such as speech quality monitoring and security of IP telephony. We sought to unify the configuration of individual components based on the NETCONF protocol. In order to be able to implement the idea into OpenWrt, we had to integrate a complex support for the NETCONF configuration protocol. Our modifications of OpenWrt in respect of NETCONF were accepted by the OpenWrt community and have been included in OpenWrt/Trunk branch. The paper explains and describes the whole concept of the BESIP project and its individual modules.

Keywords SIP · OpenWrt · BESIP · NETCONF · Security · Management · Monitoring

1 Introduction

The Bright Embedded Solution for IP Telephony (BESIP) project was formally launched in mid-2011. Our intent was to develop a modular architecture for a SIP communication server with additional functionalities such as speech quality monitoring and protection against certain security threats. The solution we developed consists of a SIP server running on OpenWrt OS with key components integrated. The entire system is distributed as an image is suitable for embedding hardware with low computational complexity. Alternatively, individual packages can be installed from SVN. Every software solution includes its own configuration and management. Accordingly, BESIP aims to be a scalable solution with an emphasis on security and a unified configuration [1].

M. Voznak (✉) · J. Slachta · L. Macura · K. Tomala
Department of Telecommunications, VSB-Technical University of Ostrava, 17. listopadu 15/2172, 708 33 Ostrava, Czech Republic
e-mail: voznak@ieee.org

J. Slachta
e-mail: jiri.slachta.st1@vsb.cz

L. Macura
e-mail: macura@opf.slu.cz

K. Tomala
e-mail: karel.tomala@vsb.cz

2 State of the art

As mentioned in the introduction, we discuss the implementation of a SIP communication server solution which would be an alternative to several current implementations. The main advantage of our solution is the ability to easily and quickly set up a full featured PBX on almost any hardware. We can presume that almost all implementations are based on open-source Asterisk PBX, web-interface for Asterisk and with a GNU/Linux distribution on the base layer.

At present, there are several projects that offer multipurpose IP telephony solutions for embedded devices and for household or enterprise platforms. The initial project of a GNU/Linux distribution which offers an easy set-up of IP telephony in a few steps is the Asterisk@Home project. The first version of this project was released on 29 April 2005. This project integrated a web interface for Asterisk, Flash Operators Panel to control and monitor PBX in real-time and also offered a full FAX support within one bootable image for almost any x86 PC. On 3rd May 2006 the development of this project was discontinued and was replaced by its successor Trixbox. However, the development of Trixbox does not seem to continue any more. Two existing projects—AsteriskNOW [2] and Elastix [3]—now offer an alternative to Trixbox. The former, AsteriskNOW appears to be similar to Trixbox— a packed GNU/Linux distribution with Asterisk with a FreePBX web interface on top of it. The latter, Elastix, is a bit more modular. Compared to any other project, it offers a slightly more modular hierarchy to facilitate the applicability to a multiple service server. The increasing popularity of embedded devices, such as Raspberry Pi, is the reason why the Micro Elastix [4] distribution was born. However, all of those projects are either prepared for x86 machines only or for specific hardware. Micro Elastix only supports three platforms, namely PICO-SAM9G45, Raspberry Pi and MCUZONE.

None of the projects includes a security module that would offer a complete IPS and IDS system to prevent attacks against the SIP Registrar server. Also there is no module that would monitor the quality of voice calls transmitted through an integrated PBX or a module that enables NETCONF capability on them. Thanks to the portability of the OpenWrt distribution we can prepare a BESIP bootable image for almost any device.

3 Design of architecture

The BESIP architecture is depicted on Fig. 1. There are four basic modules: Core, Security, Monitoring and PBX.

First, we discussed the existing projects which we could adopt and modify for our purposes. We considered the following open-source tools and applications:

- OpenWrt for good scalability and simple embedding;
- Kamailio for reliability and high availability [5];
- Asterisk and Kamailio as Back-to-Back User Agent (B2BUA) and SIP Proxy [6];
- YUMA as NETCONF server [7] and OpenWrt UCI as configuration backend;

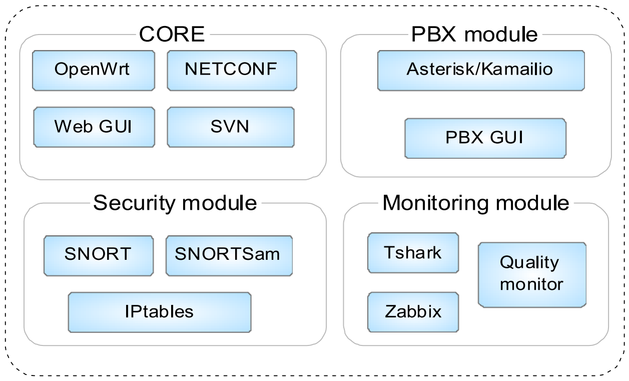


Fig. 1 BESIP architecture

- SNORT with SNORTSAM and IPtables as an intrusion detection and protection system [8].

The core is divided into the following parts: OpenWrt as build platform; NETCONF to administer the entire system, Web GUI for a user-friendly configuration and SUBVERSION as a revision control system. The security module is based on SNORT, SNORTSam and IPtables [8,9]. The monitoring module exploits a tshark package and our developed tool which provides information about the quality of a particular speech [10,11]. The Zabbix agent is used to report basic states of the entire system. Finally the PBX module is developed on Kamailio in conjunction with Asterisk. Both individual packages and the whole image for particular HW used for testing of pre-released distributions are available for download [12].

Several open-source applications were adopted and implemented into the developed modules. However many modifications were required in the implementation, especially in the OpenWrt core module. This is due to a complicated porting of applications into OpenWrt buildroot. Our patches were tested and accepted by the OpenWrt community. Having observed and discussed various open-source solutions, OpenWrt was selected as the primary platform. There are many packages included; packages not included can be added into the application tree. Even if for certain packages this is not an easy procedure, especially for packages without configure script, we opted for this solution. OpenWrt is well-known for excellent support, solid ticket system, relatively good documentation and cooperation with the community of developers.

4 Core module

The long-term goal of this project is to make the BESIP configuration independent of clients. Today, many systems are configurable using web, SSH or telnet. Each of them offers its own semantics and configuration file. The BESIP project aims to change this situation, using NETCONF as the defined communication and management protocol. In addition, configuration independent syntax will be available on all modules. During the first stage of the project, applications and libraries were ported. Afterwards, we focused on implementing NETCONF, UCI, PBX, Security and Monitoring modules. The NETCONF protocol allows the device to send and receive configuration data through XML documents using the RPC paradigm [13,14]. These XML documents are handed over the RPC calls. The RPC request is initiated by a client that requests the configuration data or a command to be performed on the server. While these requests are being performed, the client is blocked until he receives the RPC reply from the NETCONF server [7,13]. The responses consist of a configuration that is either complete or partial. Another reply

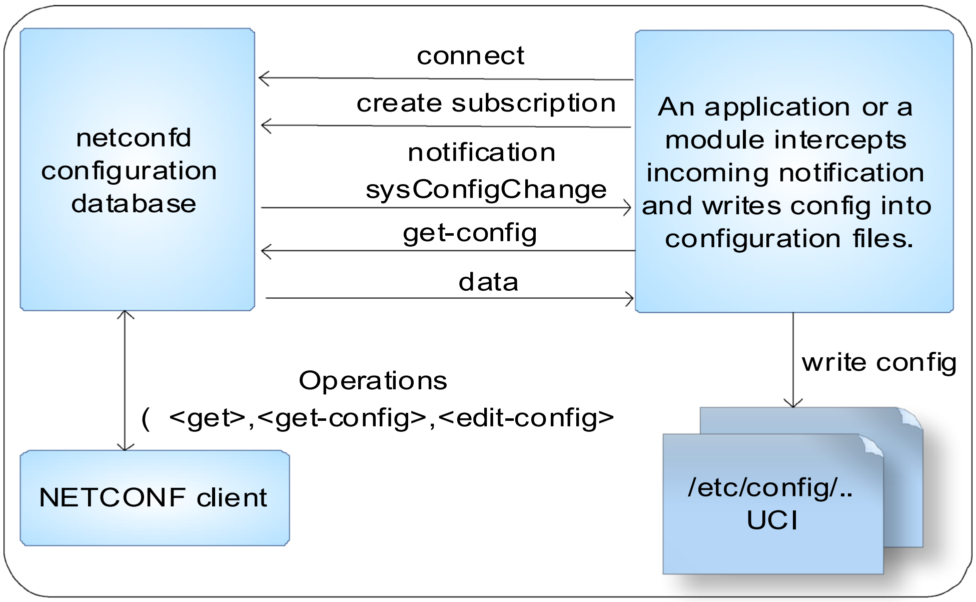

connect
create subscription
notification sysConfigChange
get-config
data
netconfd configuration database
An application or a module intercepts incoming notification and writes config into configuration files.
write config
Operations (<get>,<get-config>,<edit-config>
NETCONF client
/etc/config/.. UCI

Fig. 2 Management based on NETCONF usage

is a message informing us whether a command was successfully performed on the server. See Fig. 2 to understand the configuration data flow defined in BESIP. The structure of configuration data on the NETCONF server in the YUMA package (netconfd) is specified by the YANG module which defines the semantics and syntax of a management feature [15]. The configuration data is stored securely on the NETCONF server and all requests and responses must comply with a firmly defined structure, specified by YANG modules. Next, a global database of all the configurable parameters is required and its role is performed by the NETCONF server.

The configuration parameters are inserted by the user and stored on the NETCONF server. Consequently, the stored configuration data is accessible through simple queries. It makes the device quickly configurable, therefore a backup or a configuration restore can be performed easily and quickly. YUMA is a package which provides tools for the network management. We ported YUMA into OpenWrt successfully. It consists of NETCONF client yangcli, NETCONF server netconfd, validation tools yangdiff and yangdump and netconf-subsystem which allows us to communicate with the NETCONF server through a SSH2 subsystem. OpenWrt uses UCI as a configuration backend. It is a group of configuration files which can be read or modified by common UCI API. The NETCONF protocol is applied for the BESIP configuration. The advantage of this approach lies in particular in the exact definition of the data model; possibility to call any function through a remote procedure call; data model editing in YANG and independence on client. A RFC draft of YANG data model for interface configuration was applied to verify proposed functionalities [16]. We combined several applications and packages to achieve NETCONF's overall functionality. The libnetconf [17] library, which has been developed by CESNET as an open-source project since 2009, is a part of our implementation. The NETCONF protocol exploits a specified mechanism for OpenWrt distribution which is a platform for embedded equipment. The primary goal is to provide a suitable environment for small routers with minimum requirements on processor, flash and RAM. Although the implementation is mostly problematic due to the cross compilation; the image generation for embedded equipment is very well parameterized. We exploit this fact in our autobuild script supporting the following targets: asuswl_500gp-brcm47xx, besiphw1-x86 and tplink1043nd-ar71xx. Each of these targets represents a set of variables defining parameters for the generation of an image of particular hardware [12].

The diversity of configuration interfaces is a remarkable feature of most applications and libraries based on GNU/Linux kernels. However, almost every application or tool is configured in a different way. OpenWrt deals away with the issue how to configure more applications in one configuration tool. Unified configuration interface (UCI) is the configuration interface in OpenWrt. All packages supporting this way of configuration can read the UCI configuration data format and create their configuration files from this data. The advantage lies in the independence of individual implementation. UCI provides an interlayer between the user and the application, simplifying configuration for users and unifying API for applications. We adopted UCI as the primary database of application's configuration data, our concept is depicted in Fig. 3. UCI only defines a format of configuration directives and access to them but not their exact content or relation to each other. That depends on the user. Typically if the user modifies the name of a network

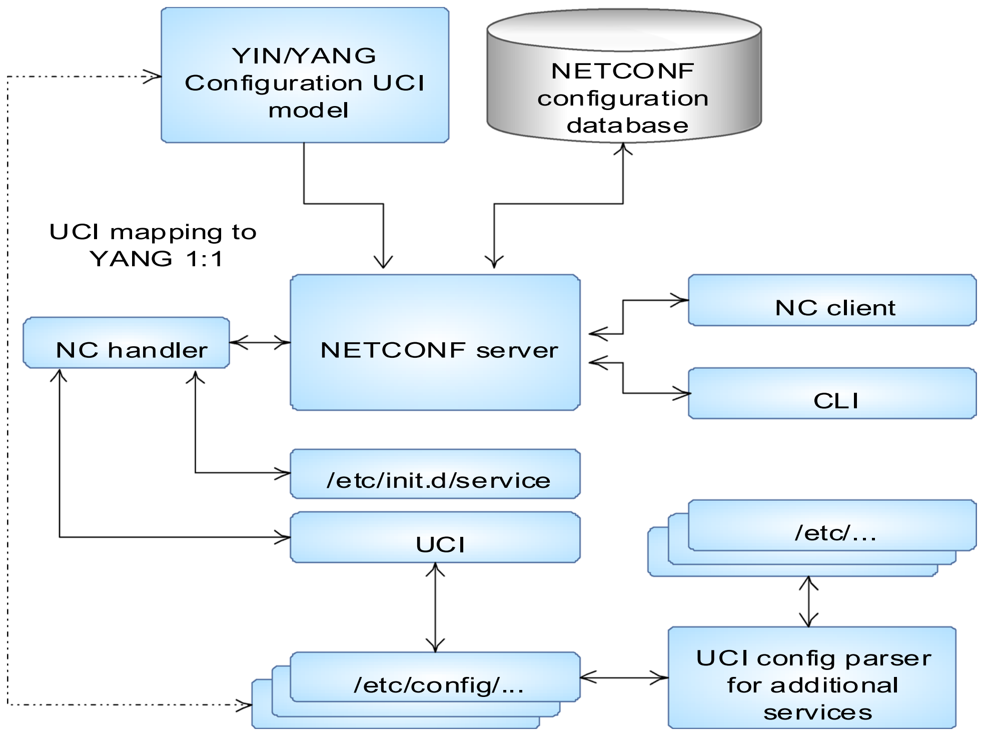


Fig. 3 Concept of NETCONF and UCI interoperability in OpenWrt

interface, the subsequent libraries will fail unless the modification is performed in all locations where it is necessary. Our approach addresses two significant issues. First issue is the imperfection and variability of the existing UCI. The second issue is a large number of scripts ensuring parsing and the formation of configurations from universal UCI files. The YANG model defines the exact configuration structure which enables to detach a substantial part of the code from current applications. If users change the UCI file, the content is automatically validated within the YANG schema and users are notified in case of a failure.

5 PBX module

The PBX module operates as SIP proxy or SIP B2BUA, depending on the configuration. Generally, the module also ensures call routing. Asterisk is used to distribute calls and performs the PBX functions. Kamailio is used to proxy SIP requests, to normalize the traffic and to ensure security [5,6]. When developing VoIP solution, there are always two factors involved. The first one is high availability and reliability, the second one is the issue of advanced functions. Many developers try to find a compromise. We implemented both and our BESIP is able to adapt to user's requirements. A more complex system can handle many PBX functions such as call recording or an interactive voice response. However, due to its higher complexity, it is more prone to fault. On the other side, SIP proxy is a simple piece of software which can perform call routing, is more fault resistant but it is also more difficult to incorporate the advanced PBX functions into it [18,19]. BESIP offers users the possibility to choose how the system will work. For this reason, BESIP includes both Kamailio and Asterisk. Today, only one of these engines can be configured but in the future, both engines will work together and will be configured by the common NETCONF server. Kamailio will be able to route requests even if Asterisk is out of order. Advanced PBX functions, however, will be unavailable.

Billing is an important feature of every SIP server. In many systems, billing is divided into two separate steps. The individual calls are processed and a call detail record (CDR) is generated for every call performed. These CDRs are stored in a text file or a repository. The second step is ensured by an application which enables to perform statistics over data stored, i.e. to search and display results according to requested criteria. This is a conventional scenario, classical approach of many billing applications and highly reliable as the PBX function is not affected by the billing. Even if there is an issue with the billing software, PBX still operates properly. There is, however, one big disadvantage: during the call setup, PBX cannot provide an authorization which is well-known from pre-paid services offered by mobile operators. Taking this into account, we are able to perform more checks and operations at the call setup level. For example, we can look at user's credit and block a call where the credit has been overdrawn or is low. Similar to this, we can authorize every call against a threshold such as maximum price per minute/trunk/global. These thresholds can be pre-set or changed dynamically according to the actual user's credit. Considering this, PBX will be safer and more resistant against attacks the aim of which is to misuse PBX [20].

6 Security module

The aim of the security module was to make the developed system as secure as possible. In case some security incident occurs, BESIP immediately addresses the situation and notifies the administrator about the event in a detailed report. The core of the entire IPS solution is the IDS system Snort which detects malicious activity in the network. The detection is based on signatures or detection of anomalies. The SnortSam application allows Snort to dynamically intervene into IPtables rules and the traffic detected is then blocked for some time. Once the attack is over and timed out, the blocked IP is allowed to communicate again. Thus, only malicious traffic that poses a threat to our server is blocked. If more soft faults appear from some IP, it is blocked at the IPtables level. This approach can effectively block incorrectly configured clients and servers. For example, if a client sends REGISTER with proper credentials, it is not an obvious security attack. But if the client attempts to register repeatedly, asking for computing sources at the SIP REGISTRAR server with every registration, such attempts can be denoted and blocked for a certain time. Security precautions against these attacks include Snort rules tracking the number of messages sent to the SIP server from a particular source address. The blocking rules were similar in most cases, like for example this Snort rule for blocking an unwanted register flood:

alert udp $EXTERNAL_NET any -> $SIP_PROXY $SIP_PORT (msg:"SIP DoS attempt(registerflood)"; content:"REGISTER sip"; detection_filter:track by_src, count 50, seconds 5; classtype:misc-attack; sid:1000001; rev:1; fwsam:src, 10 min;)

Administrators can use the Zabbix or NAGIOS agent inside BESIP to gather all information directly into their monitoring system. Monitoring is an important part of the security module. The BESIP team had focused on the issue in early stages of project designing [1]. Depending on hardware used, BESIP can resistcertain DoS attacks [21]. If hardware is strong enough to detect some security incidents on the application level, the source IP is immediately dropped. With weak hardware, this can, however, pose a serious problem. The features of our security module were verified in a test-bed and results are depicted in Figs. 4 and 5. The CPU load was monitored during trivial SIP attacks.

The Snort, SnortSam, IPtables (SSI) line is the response when the security module in BESIP is active. The following dependencies were measured without SSI. Only two types of DoS attacks were emulated, namely REGISTER flood and INVITE flood. We used sipp generator to generate the former, and we used inviteflood tool to generate the latter. The dependencies in both figures clearly prove the ability of the security module to mitigate the attacks performed.

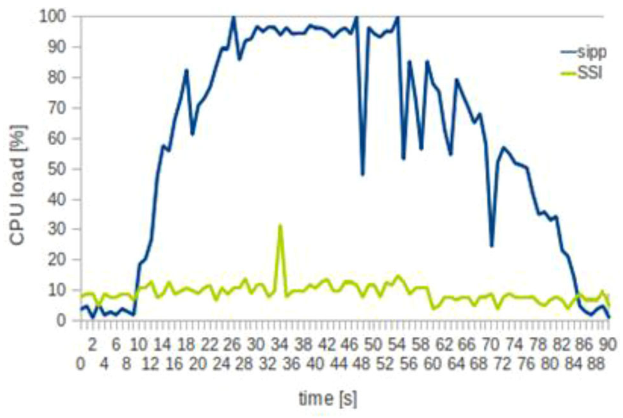


Fig. 4 Attack effectiveness based on REGISTER flood

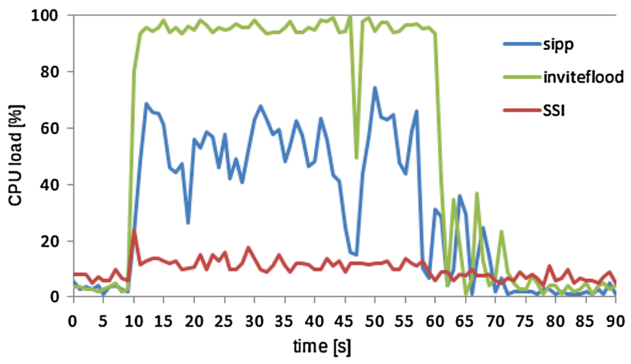


Fig. 5 Attack efficiency based on INVITE flood

7 Monitoring module

The overall solution of the monitoring system consists of several different open source components plus the part that was developed for this purpose to meet the defined requirements. We use a computational E-model [22], simplified for the purpose of the implementation. The computation can be split into several elements and is expressed by eq. (1). The used values were adopted from recommendation ITU-T G.107 where the default values are listed and finally we need to express only two uknown parameters I_d and I_{e-eff} which are described in eqs. (4) and (5).

$$R = 94.7688 - 1.4136 - I_d - I_{e-eff} + 0 \tag{1}$$

The simplified E-model only takes into account the effects from codec, packet loss and end-to-end delay. The situation is depicted in Fig. 6.

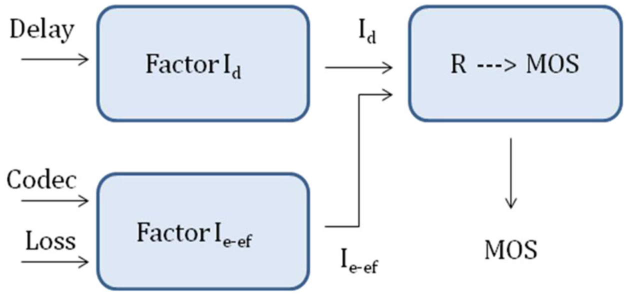


Fig. 6 E-model in a simplified version

The computed R is converted to MOS value. For this purpose, relations (2) and (3) were applied [23].

$$MOS = 1 \text{ for } R < 6.5 \text{ and } MOS = 4.5 \text{ for } R > 100 \quad (2)$$

$$MOS = 1 + 0.035 \cdot R + R \cdot (R - 60) \cdot (100 - R) \cdot 7 \cdot 10^{-6} \text{ for } 6.5 \leq R \leq 100 \quad (3)$$

Factor I$_d$ represents all impairments which are caused by different combinations of delays. We applied a linear regression to results gained in AT&T laboratories [24] and derived relation (4) which provides accurate results, with regression quality r=0.99 ranging from 0 to 400 ms.

$$I_d = \begin{cases} 0.0267 \cdot T & T < 175\,\text{ms} \\ 0.1194 \cdot T - 15.876 & 175\,\text{ms} \leq T \leq 400\,\text{ms} \end{cases} \quad (4)$$

Packet-loss dependent effective equipment impairment factor I$_{e-eff}$ is derived using the codec-specific value for the equipment impairment factor at zero packet-loss I$_e$ and packet-loss robustness factor B$_{pl}$, both listed in Appendix I of ITU-T G.113 [25]. With packet-loss probability P$_{pl}$, I_{e-eff} is calculated using the equation (5).

$$I_{e-eff} = I_e + (95 - I_e) \cdot \frac{P_{pl}}{\frac{P_{pl}}{BurstR} + B_{pl}} \quad (5)$$

BurstR is the so-called burst ratio, defined as the ratio between "Average length of observed bursts in an arrival sequence" and "Average length of bursts expected for the network under a random loss". The system itself consists of three logical components which are a web interface that serves the administrators (Web GUI), part of the script (Scripts) that controls gathering data necessary to compute the speech quality in the simplified E-model. The last component is the Quality Monitor which contains the logic for calculation itself and processes data obtained by scripts. All data is stored in a SQLite3 database. The open source tool Java API SQLiteJDBC was selected to enable communication with the database. Speech quality monitoring is controlled through a web interface that is accessible from the main crossroads management interface (the situation is depicted in Fig. 7). The web interface is written entirely in the PHP scripting

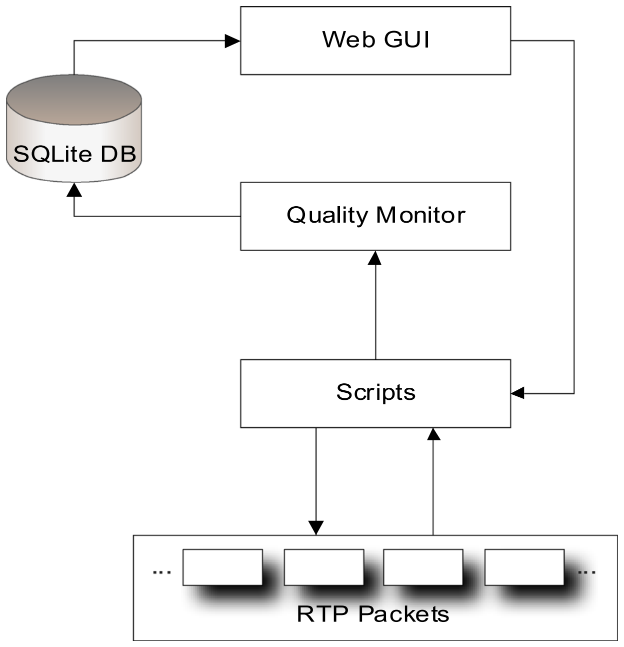


Fig. 7 Overview of the logical structure of VQM

language in order to enable starting or stopping the monitoring system through the OpenWrt shell as it depends on shell applications such as tshark (a small terminal-based network analyzer). Scripts are launched through the web interface of the monitoring tool enabling the monitoring as such. In practice, this means starting the network traffic capture with the tshark tool with the RTP filter activated. The usage of the RTP filter makes working with RTP streams much easier as these streams contain some important statistical data (packet loss, jitter) and other important information (source/destination IP, codec) necessary to calculate the speech quality in the E-model. The developed application offers the comfort of management in a web application, the developed interface aggregates required functions.

The function state diagram is depicted in Fig. 8. By clicking the Start button, the PHP script _start.php is launched at the server side using the Java script and Ajax which launches the _start.sh shell script in the background. It activates tshark. It is terminated every 900 s and data obtained from the RTP filter is written into the sterams.log file. At this stage, the recalculation is not performed after each step. The script automatically computes the parameters and clears streams.log after three predefined rotations. By clicking the Stop button, the _stop.php script is launched using the Java script and Ajax at the server side. The _stop.sh shell script terminates any tshark and sh_start.sh processes by means of the kill command. The application automatically calculates the results after three finished cycles of tshark, i.e. after 45 min. By clicking the Results button, the _perform.php script is launched using the Java script and Ajax at the server side which runs the _perform.sh shell script. Where the

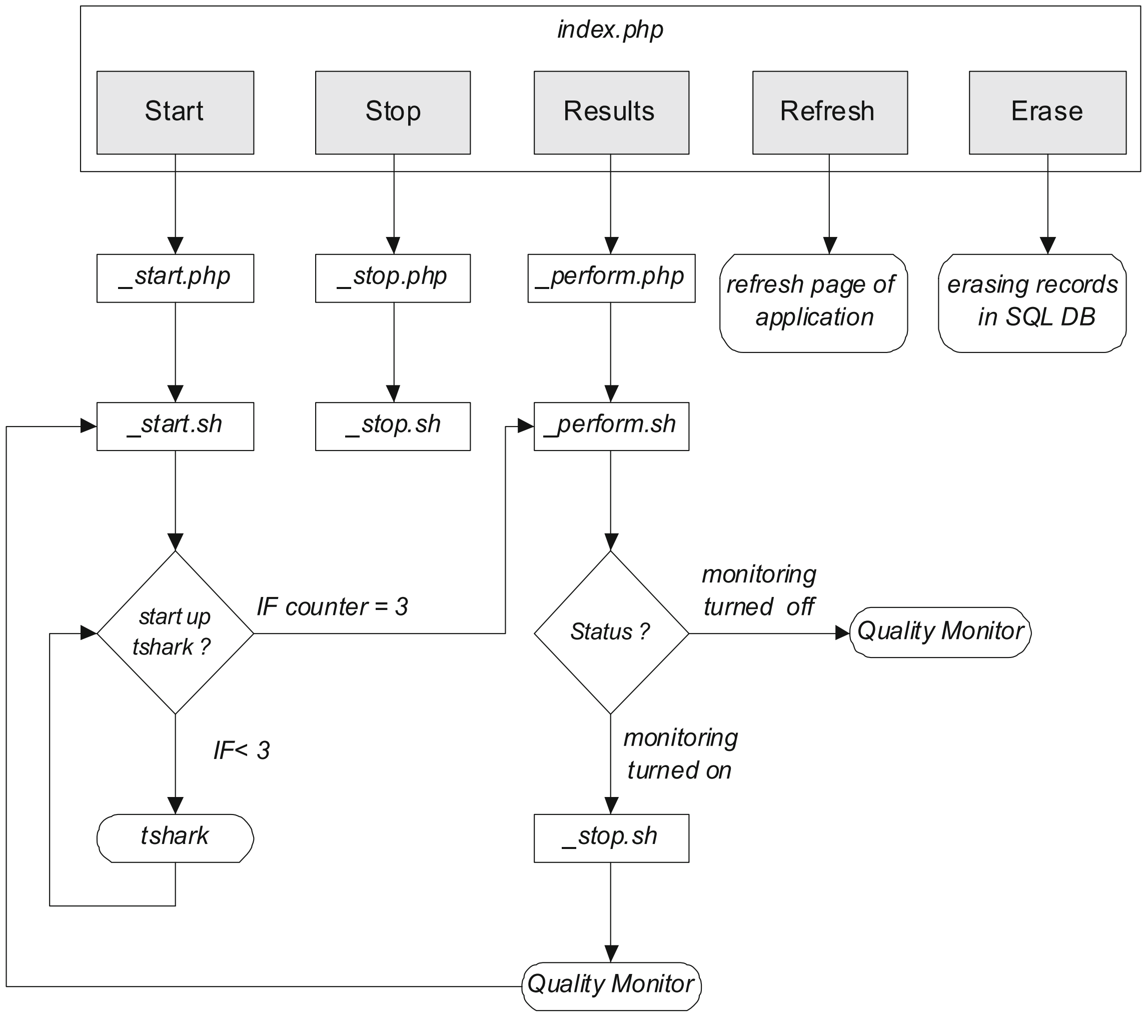


Fig. 8 The function state diagram

background monitoring is currently running, the php script first launches _stop.sh, then performs the calculation (_perform.sh) and finally resumes the monitoring using _start.sh. Where the monitoring is turned off, only the part with the the_perform.sh script is performed.

The _perform.sh shell script contains the start of Java application QualityParser written for this purpose. The contents of the file that stores information from tshark (streams.log) is deleted once the data had been processed. The Erase button deletes all records in the database. The page does not refresh automatically after the button has been pressed, it must be refreshed manually (by using the Refresh button). The function is applied to erase all records from the SQL database.

8 Performance evaluation

In order to evaluate the performance of the BESIP system, we tested our BESIP running on low-end HW platform containing x86 Intel Atom D410, RAM 1GB/677 MHz and 16 GB SSD. The performance test was carried in accordance with new standard RFC 6076 with our own benchmarking tool [26]. We found out that the tested platform with BESIP is able to handle 256 registrations per second and is stable up to 6000 registrations. Its limitation is 90 simultaneously running calls. When this limit was exceeded, we observed unsuccessful calls and a rapidly increasing SRD (Session request Delay);for example75 % rate of successfully established sessions in case of 150 simultaneous calls and with SRD over 60 ms.

9 Conclusion

The output of our work is the entire BESIP concept and its implementation. As we mentioned, BESIP consists of several components which are distributed under GPL as an open-source solution. Several of them, for example the components in Security and PBX modules, were adopted fully; some of them, like for instance the Core module, in a modified version. Finally we developed our own tool for Speech quality assessment. The contribution of our work is not only the coding of the BESIP system; we also conceived the idea of the unified configuration management with unified CLI syntax which enables to configure different systems (in our case Asterisk and Kamailio). BESIP is distributed as a functional image for x86 platform but can be run on Vmware or KVM. Its configuration is available through a web browser or SSH client [12].

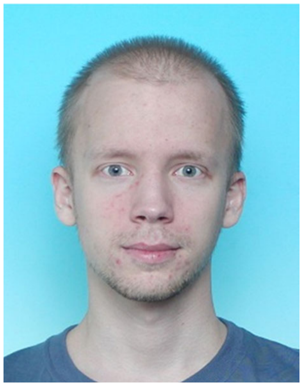
We noted that we still need to solve many issues. Individual packages are working, version 1.0 was released after several pre-releases in November 2011 and version 2.0 will be released in mid-2013. New release 2.0 will be based entirely on NETCONF with one API to configure the entire system. Further, CLI syntax is being developed and will be connected to NETCONF. CLI will be independent of internal software so when any internal software is modified, there will be no need to make changes in the configuration. In addition, CLI and NETCONF configuration will be independent of hardware and version. To export a configuration from one box and to import it to the next one would be a simple task. Users will only need to modify a single configuration file to manage the entire box. After this step, all internals of configuration will be hidden as was mentioned in the introduction. All BESIP binary images from nightly autobuild can be downloaded from [12].

Acknowledgments This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence Project (CZ.1.05/1.1.00/02.0070) and by the Development of human resources in research and development of latest soft computing methods and their application in practice Project (CZ.1.07/2.3.00/20.0072) funded by Operational Programme Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic.

References

1. Macura, L., Voznak, M. (2012). Embedded multiplatform SIP server solution. In *35th International Conference on Telecommunication and Signal Processing, TSP2012* (pp. 263–266).
2. AsteriskNOW, a complete Linux distribution with Asterisk, the DAHDI driver framework, and, optionally, the FreePBX administrative GUI. http://www.asterisk.org/downloads/asterisknow.
3. LANDIVAR, Edgar. (2011). Unified Communications with Elastix, Vol. ASIN B0056IXJRC.
4. uElastix, embedded variant of Elastix distribution. http://uelastix.org/.
5. Madsen, L., Meggelen, J., & Bryant, R. (2011). *Asterisk: The defnitive guide*. Sebastopol: O'Reilly Media.
6. Goncalves, F. (2008). *Building telephony systems with OpenSER*. Birmingham: Packt Publishing.
7. Enns, R., et al. (2011). Network Configuration Protocol (NETCONF), IETF RFC 6241.
8. Voznak, M., & Safarik, J. (2012). DoS attacks targeting SIP server and improvements of robustness. *International Journal of Mathematics and Computers in Simulation*, *6*(1), 177–184.
9. Voznak, M., Rezac, F., Tomala, K (2010). SIP Penetration test system. In *34th International Conference on Telecommunications and Signal Processing, TSP2010* (pp. 497–503).
10. Cano, M., & Cerdan, F. (2012). Subjective QoE analysis of VoIP applications in a wireless campus environment. *Telecommunication Systems*, *49*(1), 5–15.
11. Jelassi, S., Youssef, H., Hoene, C., & Pujolle, G. (2012). Single-ended parametric voicing-aware models for live assessment of packetized VoIP conversations. *Telecommunication Systems*, *49*(1), 17–34.
12. Source code of BESIP Project (2012), LipTel Team. http://liptel.vsb.cz/mirror/besip/nightly
13. Cutuli, G., Mumolo, E. Tessarotto, M. (2002). An XML-based virtual machine for distributed computing in a For/Join framework. In *24th International Conference Information Technology Interface* (pp. 471–477).
14. Chisholm, S., Trevino, H. (2008). NETCONF Event Notifications, IETF RFC 5277.
15. Scott, M. Bjorklund, M. (2010). YANG Module for NETCONF Monitoring, IETF RFC 6022.
16. Bjorklund, M. (2010). YANG—A data modeling language for the network configuration protocol (NETCONF), IETF RFC 6020.
17. Libnetconf, NETCONF library in C. http://code.google.com/p/libnetconf.
18. Chochelinski, R., Baronak, I. (2009). Private telecommunication network based on NGN. In *32nd International Conference on Telecommunications and Signal Processing* (pp. 162–167).
19. Simoes, J., & Magedanz, T. (2011, December). Contextualized user-centric multimedia delivery system for next generation networks. *Telecommunication Systems*, *48*(3), 301–316.
20. Rezac, F., Voznak, M., Tomala, K., Rozhon, J., & Vychodil, J. (2011). Security analysis system to detect threats on a SIP VoIP infrascructure elements. *Advances in Electrical and Electronic Engineering*, *9*(5), 225–232.
21. Sisalem, D., Kuthan, J., & Ehlert, S. (2006). Denial of service attacks targeting a SIP VoIP infrastructure: Attack scenarios and prevention mechanisms. *IEEE Network*, *20*(5), 26–31.
22. ITU-T. (12/2011). The E-model: A computational model for use in transmission planning, Recommendation ITU-T G.107. International Telecommunication Union, (pp. 10).
23. ITU-T. (2011). The E-model: A computational model for use in transmission planning, Recommendation ITU-T G.107. International Telecommunication Union.
24. Cole, G., & Rosenbluth, H. (2001). Voice over IP performance monitoring. *ACM SIGCOMM Computer Communication*, *31*(2), 9–24.
25. ITU-T. (2007). Transmission impairments due to speech processing, Recommendation ITU-T G.113. International Telecommunication Union.
26. Rozhon, J., & Voznak, M. (2011). SIP registration burst load test. *Communications in Computer and Information Science*, *189*(2), 329–336.

Miroslav Voznak is an Associate Professor with Dpt. of Telecommunications, Technical University of Ostrava. He received his M.S. and Ph.D. degrees in telecommunications, dissertation thesis "Voice traffic optimization with regard to speech quality in network with VoIP technology" from the Technical University of Ostrava, in 1995 and 2002, respectively. Topics of his research interests are Next Generation Networks, IP telephony, speech quality and network security. He was involved in several FP EU projects.



Jiri Slachta is a M.S. student with Department of Telecommunications at Faculty of Electrical Engineering and Computer Science, VSB- Technical University of Ostrava. His professional activities are focused on Embedded systems, Networks and Application development for mobile systems. He received dean's appreciation for his bachelor thesis in 2011.

Lukas Macura is a Ph.D. student with Dpt. of Telecommunications at Faculty of Electrical Engineering and Computer Science, VSB- Technical University of Ostrava where he finished his bachelor and Master of Science studies. His professional and research domain is networking protocols and network management applications.

Karel Tomala is a Ph.D. student with Dpt. of Telecommunications at Faculty of Electrical Engineering and Computer Science, VSB- Technical University of Ostrava where he received his M.S degree in telecommunications. His research is focused on new methods of speech quality assessment and he takes his professional interest in Networks and Application development.