Miroslav VOZNAK, Karel TOMALA, Jiri VYCHODIL, Jiri SLACHTA

VSB-Technical University of Ostrava, Czech Republic

Advanced Concept of Voice Communication Server on Embedded Platform

***Abstract**. The paper deals with a design of an embedded Voice communication server which was developed within the scope of the BESIP project (Bright Embedded Solution for IP Telephony). The project brings a modular architecture with additional functionality such as a speech quality monitoring and a protection against security threats.The speech quality assessment is carried out in a simplified computational E-model and we implemented our proposal into the BESIP as an optional component. In the security module. We applied a standard approach to the intrusion detection and protection and in addition to the mentioned modules we come up with an idea of unified configuration based on the NETCONF protocol. We implemented ntegrated the complex support of NETCONF configuration protoco into OpenWRT and our modifications were accepted by OpenWRT community. The paper describes the inidvidual modules, their features and entire BESIP concept.*

***Streszczenie.** W artykule przedstawiono budowę serwera komunikacji głosowej, jako część projektu BESIP. Opracowano architekturę modułową, posiadającą dodatkowe funkcjonalności, takie jak: kontrola jakości dźwięku mowy i ochrona przed zagrożeniami zewnętrznymi. Zastosowano ideę konfiguracji jednolitej, opartej na protokole NETCONF. Opisany został każdy z modułów, ich funkcjonalność i cały projekt BESIP. (**Ulepszona koncepcja serwera komunikacji głosowej na platformie osadzonej**).*

Keywords: BESIP, NETCONF protocol, OpenWRT, Speech quality, VoIP security.
Słowa kluczowe: BESIP, NETCONF, OpenWRT, jakość mowy, ochrona VoIP.

Introduction

Our intent in the BESIP project (Bright Embedded Solution for IP Telephony) was focused on development an open-source modular architecture of voice communication server with additional functionalities such as speech quality monitoring and protection from selected security threats. The BESIP offers the prepared solution with integrated key components, the entire system is distributed as an image or individual packages can be installed from SVN and the projetcs aims to be scalable solution with security and unified configuration in mind [2].

First, we discussed existing open-source projects which we could adopt and modified for our purposes, we took into account several tools and applications such as OpenWRT for good scalability and simple embedding [1], Kamailio for reliability and high availability [2], Asterisk and Kamailio as B2BUA (Back-to-Back User Agent) and SIP Proxy [3], YUMA as NETCONF server [4], OpenWRT UCI (Unified Configuration Interface) as configuration backend and finally SNORT as an intrusion detection and protection system [5].

Several open-source applications were adopted and implemented into developed modules however within the implementation many modifications were required, especially in the core module based on OpenWRT due to complicated porting of applications into OpenWRT buildroot. Our patches were verified and accepted by OpenWRT community. The speech quality monitoring tool was developed from scratch and implemented in Java. BESIP can run on low-end hardware such as ARMv5/400 MHz with 32MB RAM and supports OpenWRT architecture ARM, MIPS, MIPSEL or x86.

The most important step which had to be done was choosing the right software distribution/platform. There was an idea to modify Debian distribution, this is probably the easiest way for developers. Debian includes many ports and packages but is not suitable for embedding. A modification of Debian, in order to be easily embedded into small device with read-only flash, is really a difficult task and the expected results of such work can not lead to a source distribution. Next solution was adopting some low-level distribution for embedding. There are several possibilities like FreeWrt, OpenWrt, DebWrt etc. After discussion and projects observations, OpenWrt was selected as primary platform. OpenWrt is well-known for great support, ticket system, relatively well documentation and cooperation with community of developers.

Proposed Architecture and Technology Used

The BESIP architecture is depicted in Fig. 1, there are four basic modules: Core, Security, Monitoring and PBX. The core is divided into following parts: OpenWRT as a build platform; NETCONF for administration of entire system with YUMA implementation; Web GUI for user-friendly configuration and SUBVERSION as revision control system providing a support and better orientation for developers.

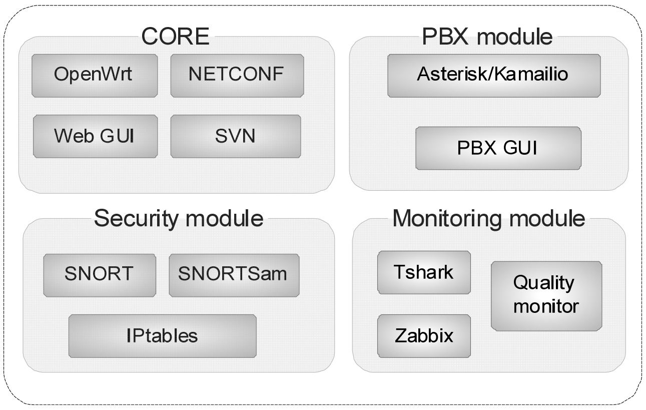


Fig. 1. BESIP architecture

The security module is based on SNORT, SNORTSam and IPtables [5]. In addition to this, the Kamailio rate limit and pike module is used for defending attacks. The monitoring module exploits a tshark package and our java code which interprets its results and gives information about particular speech quality. The Zabbix agent is used to report basic states of entire system and finally the PBX module is made from Kamailio in conjunction with Asterisk.

As for the distribution, not only individual packages are available for download but the whole image for particular HW used for testing of pre-released distributions such as HW depicted in Fig. 2 can also be downloaded [7].

Fig. 2. Suitable HW platform containing x86 Intel Atom D410 1.66 GHz, RAM 1GB/677MHz and 16GB SSD

Core Module

The long term goal of this project is to make the BESIP configuration independent on clients. Today, many systems are configurable using web, ssh or telnet and each of them offers its own semantics and configuration files. The BESIP project aims to change this situation, using NETCONF as defined communication and management protocol, configuration independent syntax will be available on all modules. The NETCONF protocol exploits a specified mechanism for exchanging the configuration data among an administrator and network devices. This protocol allows the device to send and receive configuration data through XML documents using the RPC paradigm [8]. 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 NETCONF server. The responses consist of a configuration that is complete or partial. Another reply is a message informing us if a command was successfully performed on the server or not. This communication is transferred over a transport protocol which has to be secured and to ensure an authentication and authorization. Most probable and secure way, how to communicate with the NETCONF server, is to use SSH2 protocol (RPC calls over SSH subsystem).

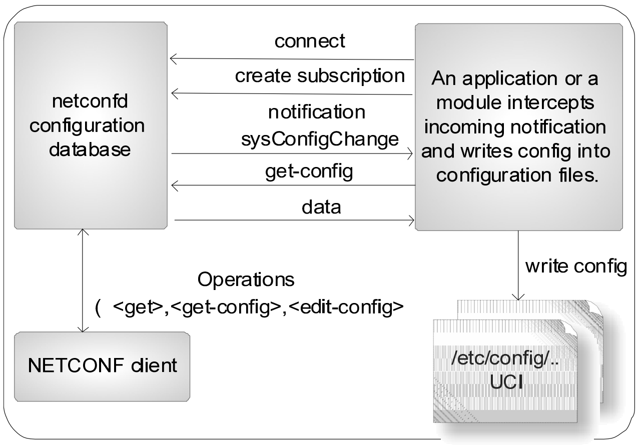

netconfd configuration database
connect
create subscription
notification sysConfigChange
get-config
data
An application or a module intercepts incoming notification and writes config into configuration files.
write config
Operations (<get>,<get-config>,<edit-config>
NETCONF client
/etc/config/.. UCI

Fig. 3. NETCONF usage

The configuration data flow, which has been defined in BESIP, is depicted in Fig.3. The structure of configuration data on NETCONF server in YUMA package (netconfd) is specified by the YANG module which defines the semantics and syntax of a management feature [9]. It provides complex data structures which allow design any data structures that will meet the requirements of developers.

The configuration data are stored securely on the NETCONF server and all requests and responses must comply with firmly defined structure, specified by YANG modules. Next, global database of all the configurable parameters is required and it is ensured by NETCONF server. The configuration parameters are inserted by the user and stored in the NETCONF server. Consequently, the stored configuration data are available through simple queries. It makes the device quickly configurable, therefore a backup or a restore of configuration can be simply and quickly performed. The YUMA is a package which provides tools for the network management, we ported successfully YUMA into OpenWRT. It consists of a NETCONF client yangcli, the 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 configuration backend, it is a group of configuration files which can be read or modified by common UCI API. We decided to provide a glue between NETCONF and UCI. The NETCONF protocol is applied for BESIP configuration, the advantage of this approach lies especially in the exact definition of data model; possibility to call any function through a remote procedure call; possibility of data model editation in YANG and the independance on client.

A RFC draft of YANG data model for interface configuration is applied for verification of basic proposed functionalities [9]. It enables to set up IP parameters of general network interfaces in any system and forms fundamentals for a development of individual YANG modules which have to be defined for UCI configurations. We combine several applications and packages for overall functionality of NETCONF. The library libnetconf, which has been developed in CESNET (Czech Educational and Science Network association) as an open-source project since 2009, is a key part of our implementation [10].

The NETCONF protocol exploits a specified mechanism for OpenWrt is a platform for embedded equipments and the primary goal is to provide a suitable environment for small routers with minimum requirements on processor, flash and RAM. Any ported application into OpenWRT has to comply with mentioned requirements above and its code is rigorously checked by openWRT community before is accepted. We adopted OpenWRT as a platform for creation of images with clear functionalities and versioning, our generated images can be used as a firmware for various devices and as a disk image for KVM or VMWARE virtualization machines.

Although the implementation is mostly complicated due to a cross compilation, the image generation for embedded equipment is very well parameterized and we exploit this feature in our autobuild script supporting following targets: asuswl500gp-brcm47xx; besiphw1-x86 and tplink1043nd-ar71xx

Each of these targets represents a set of variables defining parameters for an image generation of particular hardware. Diversity of configuration interfaces is a remarkable feature of most applications and libraries based on GNU/Linux kernels. Each application or tool is mostly configured in different way, this issue, how to configure more applications in one configuration tool, is solved in OpenWRT.

Unified Configuration Interface is configuration interface (UCI) in OpenWRT, all packages supporting this way of configuration are able to read configuration data form UCI and create their configuration files from these data. The advantage lies in independence of individual implementation, UCI provides an interlayer between user and application which brings a simplification of configuration for users and the unified API for applications.

The UCI only defines a format of configuration directives and access to them but no their exact content or relation each other. It depends on user and typically, if users modify a name of network interface, the next libraries fail until the modification is performed in all locations where is necessary.

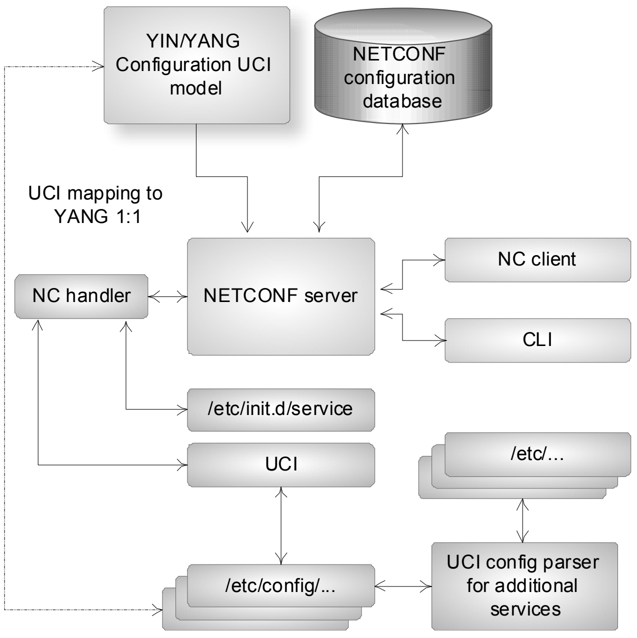


Fig. 4. Concept of NETCONF and UCI interoperability in OpenWRT

In contrast to UCI, the NETCONF and YANG embody exactly defined features. Each configuration directive has to be defined and described in YANG model. The relations among modules and options are described in YANG as well and UCI ensures only checking of data syntax. Our aim is to define and develop YANG models for individual UCI configurations, next to this to specify all dependencies and to determine ranges of possible values, e.g. in configuration of firewall we are able to force that the name of network interface corresponds to the name of module „interfaces“. Important advantage of our approach is the fact, that the verification is performed before data processing by application, this way the complex procedures in applications are detached from input data verification. Data are validated at level of NETCONF and end-user can set only parameters and values which are defined within YANG schema, the situation is depicted in Fig. 4.

Our approach solves two significant issues, the first one is an imperfection and variability of current UCI documentation, nowadays an edited WIKI but in future YANG definitions enabling an automatic generation of web content, the second issue is a large number of scripts ensuring parsing and a formation of configurations from universal UCI files. The YANG model exactly defines configuration structure which enables detach substantial part of code from current applications. If users change UCI file, the content will be automatically validated within YANG schema and users notified on failures.

UCI enables a description of configurations in OpenWRT and implementation can be realized by means: script shell function, C library libuci or UCI command line. The most of packages have in use the script shell functions to load options and consequently to generate configuration files for individual packages.

PBX Module

The PBX module is key part of the BESIP project. It operates as SIP proxy or SIP B2BUA, depending on configuration, and ensures a call routing. Asterisk is used for call manipulation and for the PBX functions. Kamailio is used for the proxying SIP requests, the traffic normalization and for the security [5]. There are always two factors when developing VoIP solution, the first one is high availability and reliability, the second one is an issue of advanced functions. Many developers try to find a compromise, we have implemented both and our BESIP is able to adapt to the users requirements. More complex system can handle many PBX functions such as a call recording or an interactive voice response but due to the bigger complexity, it is more susceptible to fault. On the opposite side, pure SIP proxy is easier software which can perform call routing, more fault tolerant but it is more difficult to use the advanced PBX functions [11]. The BESIP offers users an option to choose how system will work. From this reason, the BESIP includes both Kamailio and Asterisk. Today, only one of these engines can be configured but in future, both engines will work together and will be configured by common NETCONF server. Kamailio will route requests even if Asterisk will be out of order, only advanced PBX functions will be unavailable in such situation.

Asterisk-GUI is very flexible web solution of Asterisk management, even if it is not NETCONF based, Asterisk-GUI was added to the first BESIP release. The reason was that at this time, there was not completed an interoperability between NETCONF and Asterisk. It is available in the next release and the implementation involved very complex task. The users can decide to use easy Asterisk-GUI for PBX setup at initial version of BESIP. Nevertheless in future version we would like to remove the Asterisk-GUI package from BESIP image and the configuration will be accessible only through new developed NETCONF based management. During implementation, we solved several technical issues concerning Asterisk-GUI in OpenWRT environment and finally we made a decision on disuse Asterisk-GUI in BESIP roadmap.

SIP Proxy Kamailio is the second tool in PBX module, Kamailio configuration is well-known due to high complexity, our effort was focused on simplification the configuration in BESIP. The original Kamailio configuration is a script which is initiated with every SIP request. A rewriting of all configuration file into UCI is not possible nevertheless in recent version of Kamailio is enabled a conditional compilation of the code and a definition of global variables. It significantly simplifies situation in case of configuration modification therefore we decided to divide Kamailio script file into several logical parts. Global definition of variables is carried out at beginning of running script and afterwards the remaining part of configuration is loaded. We are able to set in UCI the basic Kamailio directives, such as option whether BESIP works as REGISTRAR server, if supports authentication, NAT, if is used as Media or RTP Proxy, etc. Our init script ensures proper distribution of parameters form UCI into Kamailio configuration

An accounting is important part of every PBX and the same way as in many systems the accounting is divided into two separate parts. The individual calls are processed in PBX modeule and a call detail record (CDR) is generated to every performed call, these CDRs are stored in text file or a repository. The next part of the accounting is an application which enables to perform statistics over stored data, it means to search and display in accordance with requested criteria. This is a conventional scenario, classical approach of many accounting applications and highly reliable because the PBX function is not affected by accounting and even if there is problem with accounting software, PBX still operates properly. However there is one big disadvantage, during a call setup, the PBX knows nothing about call price and cannot provide an authorization which is well-known from pre-paid services offered by mobile operators. Having this information, we are able to perform more checks and operations at the call setup level. For example, we can look up into user credit and do not permit a call if the credit is depleted or low. Similar to this, we can authorize every call against a threshold, such as maximal price per minute/trunk/global. These thresholds can be pre-set or dynamically changed according to the actual user credit. Having this information, the PBX will be safer and resistant against attacks aimed at an exploitation of the PBX [5].

Security Module

The security module is very important part of BESIP and all the time, it was considered to make the developed system as secure as possible. Next to this, entire system has to be fault-tolerant, monitored and protected from attacks. It means that if the device is under attack, only attacker has to be blocked, not entire system or other users. If there is some security incident, BESIP immediately solves the situation and notifies this event in detailed report to the administrator.

Attacks against the embedded systems are more dangerous due to their relatively lower performance which makes the attacks more efficient. We chose an IPS system, consisting of three applications. The core of the entire IPS solution is IDS system Snort which detects malicious activity in the network. The detection is based on signatures or detection of anomalies. The whole IDS system is modular, consisting of the following components: Packet decode capturing packets from network interfaces, Pre–processor preparing packets before the processing; Detection engine; Logging system and finally Output modules or plugins for adding another features.

The second part a SnortSam application operates on the client–server model. It allows Snort to dynamically intervene into IPtables rules. To ensure its proper operation, we need to first patch our Snort installation with a SnortSam plugin. The client communicates with the Snort's sensor, sends commands to the server (when incident has been detected). The server listens on port 898, applying information from clients to IPtables rules. SnortSam messages are transferred as encrypted, based on preshared passwords which must be same on server and on client A whitelist of non–blockable IP addresses is also available.

The detected traffic 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. The thirds part of Security modele is created by IPtables, the tool represents an open–source firewall for Linux–based operation systems. It is used to block malicious traffic on a server. In our case, running at the same physical device as a VoIP server.

The attack are recognized and processed by SNORT rules, the source IP address is automatically sent into firewall by SNORTSam and the intruder's IP is blocked. This is very flexible, reliable and effective implementation. Dropping attack based on IP directly in the Linux kernel is much more efficient than to check messages on the application level. Only first messages are going to SNORT filter. When SNORT identifies a suspicious traffic, next messages from the same IP are blocked. In next BESIP releases, we are going to to implement ipqbdb mechanism which will be even more self-defending. It is based on IP denoting.

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 client sends REGISTER with proper credentials, it is not obviously security attack but the client attempt to register again and again, with every registration requires computing sources at SIP REGISTRAR server. Such attempts can be denoted and blocked for a time interval. 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 this Snort rule for blocking 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,
10min;)
```

Administrators can use Zabbix or NAGIOS agent inside BESIP to gather all information directly into their monitoring system. The monitoring is very important part of the security module and BESIP team was already focused on the issue in early design [1]. Partially, BESIP is resistant to some kind of DoS attacks. It depends on hardware used. If hardware is strong enough to detect some security incidents on application level, the source IP is immediately dropped. But for weak hardware it can be serious problem. In such case, it is better to stop DoS attacks before it reaches BESIP. For example, SNORT on a dedicated machine will be much more flexible than if is an integral part of VoIP system. Therefore, we recommend to use an external IPS system to make VoIP service robust and secure. Nevertheless BESIP includes own IPS/IDS system. The efficiency of our security module was verified in test-bed and the achieved results in REGISTER flood are depicted in Fig. 5.

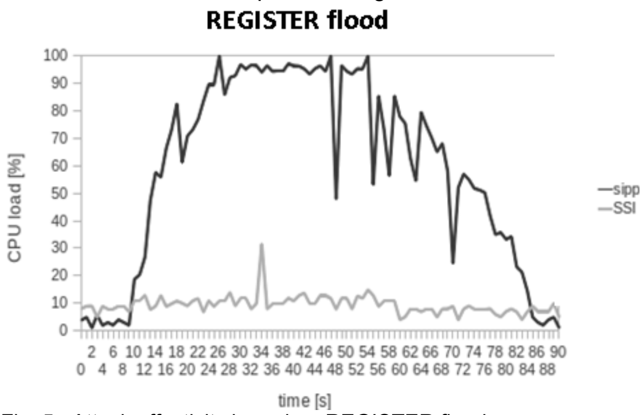


Fig. 5. Attack effectivity based on REGISTER flood.

The CPU load was monitored during trivial SIP attacks and in order to generate these attacks, we used sipp generator, the CPU indicated heavy load. On the other hand, the line SSI (Snort, SnortSam, IPtables) represents the response in case of active security module in BESIP. The dependence in Fig. 5 clearly proves the ability of security module to mitigate the performed attacks.

Monitoring Module

The overall solution of the monitoring system consists of several different open source components and also of the part that was directly developed for this purpose to meet the defined requirements. At first we deal with the application of the computational E-model, simplified for the purpose of implementation. The computational model consists of various mathematical operations over all parameters of the transmission system [12, 13, 15]. The computation itself can be split into several elements and is expressed by the following equation (1):

(1) $$R = R_o - I_s - I_d - I_{e-eff} + A$$

where R_0 represents the signal-to-noise ratio and includes all types of noise, such noises caused by the device's electrical circuit and noises arisen on the wiring. The impairment factor I_S comprises all possible impairments combinations that appear more or less simultaneously with a useful voice signal. The factor I_d represents all impairments which are caused by different combinations of delays and the factor I_{e-eff} comprises impairments caused by using a particular voice codec, occurrence of packet loss and its resistance against losses. Specific impairment factor values for codec operation under random packet-loss have formerly been treated using tabulated, packet-loss dependent Ie values. Now, the packet-loss robustness Factor B_{pl} is defined as a codec-specific value. The 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 Ie and the packet-loss robustness factor B_{pl}, both listed in Appendix I of ITU-T G.113 for several codecs [18]. With the packet-loss probability P_{pl}, I_{e-eff} is calculated using the equation (2).

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

BurstR is the so-called burst ratio, defined as ratio between "Average length of observed bursts in an arrival sequence" and "Average length of bursts expected for the network under random loss". Finally, parameter *A* slightly adjusts the final quality depending on user's concentration. The value of conventional (wire-bound) communication system is A=0, mobility by cellular networks in a building A=5, mobility in a geographical area or moving in a vehicle A=10 and access to hard-to-reach locations, e.g. via multi-hop satellite connections A=20. It should be noted that the above values are only provisional. The use of factor *A* and its selected value in a specific application is up to the planner's decision. Additional background information on the advantages of factor *A* can be found in Appendix II to ITU-T G.113.

The simplified E-model takes into account only effects from codec, packet loss (random packet loss) and end-to-end delay. Fig. 6 illustrates the situation which corresponds to relation (4).

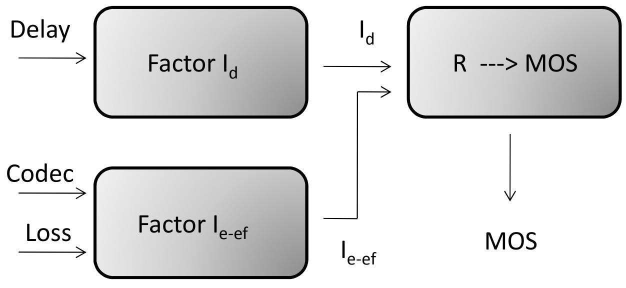


Fig. 6 E-model in simplified version

As for the codec, it is simply identified at the receiving side. The same applies to the delay. We applied a linear regression to results gained in AT&T laboratories [14] and derived relation (3) which provides accurate results, with regression quality r=0.99 ranging from 0 to 400 ms.

(3) $$I_d = \begin{cases} 0.0267 \cdot T & T < 175\text{ms} \\ 0.1194 \cdot T - 15.876 & 175\text{ms} \le T \le 400\text{ms} \end{cases}$$

Parameters R_0, I_S and *A* are replaced by constants, with their values stated in recommendation ITU-T G.107. The original relation (1) has been modified as follows (4):

(4) $$R = 94.7688 - 1.4136 - I_d - I_{e-eff} + 0$$

Parameter I_{e-eff} is computed in relation (2). Where the packet loss distribution is unknown, the value of the packet loss is assumed as random and BurstR = 1 and it results in the following simplification. Parameter I_e is fully taken over from recommendation ITU-T G.113 where its values for the most used codecs are listed [13].

Finally, the computed R-factor is converted to MOS value. For this purpose, relation (5) was adopted [15]. MOS values > 100 can be achieved only provided a wide-band codec is used.

(5) $$\begin{aligned} &MOS = 1 \\ &\quad \text{for } R < 6.5 \\ &MOS = 1 + 0.035 \cdot R + R \cdot (R - 60) \cdot (100 - R) \cdot 7 \cdot 10^{-6} \\ &\quad \text{for } 6.5 \le R \le 100 \\ &MOS = 4.5 \\ &\quad \text{or } R > 100 \end{aligned}$$

Implementation

System structure is depicted in Fig. 7. The system itself consists of three logical components, which are – web interface that serves the administrators (Web GUI), part of the script (Scripts) that controls the obtaining the information necessary to compute the speech quality in the simplified E-model. Last component is part of the Quality Monitor, which contains the logic for calculation itself and performs processing of data obtained by scripts. In the overview SQLite3 database, which is used to store the results.

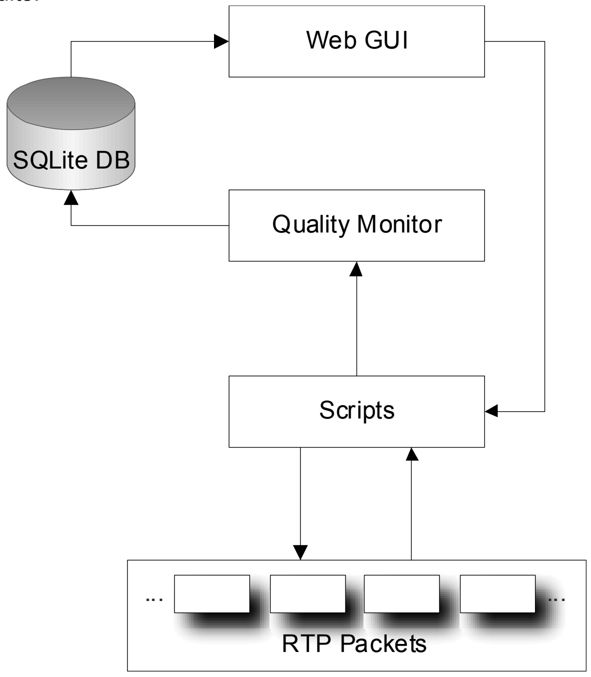
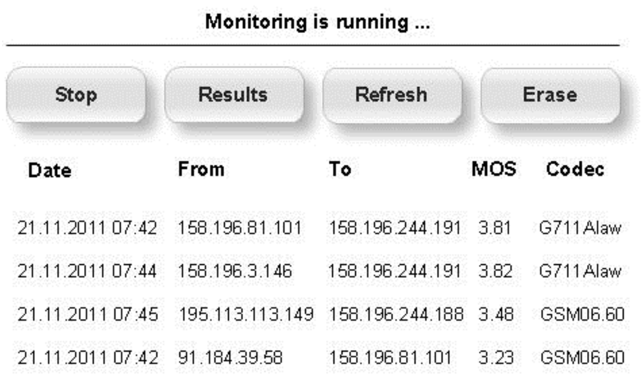


Fig. 7 Overview of the logical structure of VQM

The developed application offers the comfort of management in a web application, the developed interface aggregates required functions. Web interface is the main part of user interaction with a monitoring tool. Monitoring tool is turned off in the default configuration and can be enabled using the intuitive main interface of BESIP any time. This part of the monitoring tools is also used as a mean to display the measured and computed results. Structure of the presented data is as follows: Time, Source IP, Destination IP, MOS and used Codec. An example of user interface is shown in Fig. 8.



Date	From	To	MOS	Codec
21.11.2011 07:42	158.196.81.101	158.196.244.191	3.81	G711Alaw
21.11.2011 07:44	158.196.3.146	158.196.244.191	3.82	G711Alaw
21.11.2011 07:45	195.113.113.149	158.196.244.188	3.48	GSM06.60
21.11.2011 07:42	91.184.39.58	158.196.81.101	3.23	GSM06.60

Fig. 8 Sample of web GUI of monitoring speech quality

The web interface is written entirely in PHP scripting 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 itself. 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 status indicator is located at the top of the GUI and indicates whether the monitoring is activated in BESIP (Monitoring is running…) or is currently turned off.

Conclusion

The contribution of our work is entire BESIP concept and its implementation. As we have mentioned, BESIP consists of several components which are distributed under GPL as an open-source solution. A few of them have been fully adopted such as the components in Security and PBX modules, some of them modified, concerning the CORE module and finally we have developed own tool for Speech quality assessment. The contribution of our work is not only few hundreds of hours spent on the development, on the coding BESIP system, we bring a new idea of the unified configuration management, with unified CLI syntax which enables to configure different systems, Asterisk and Kamailio in our case. We perceive that we need to solve a lot of issues, After several pre-releases, the first stable version 1.0 was released in November 2011, the current version 1.2 is on-line available as open-source [6]. The BESIP is distributed as a functional image for x86 platform but is possible to run it on Vmware or KVM. Configuration is available through web-browser or SSH client.

As for future work, we develop a new release 2.0 which will be based on NETCONF with one API to configure entire system. The CLI and NETCONF configuration will be independent on hardware and version. To export configuration from one box and to import it to the next one will be simple task. Users could modify only one configuration file to manage entire box. After this step, all internals of configuration will be hidden as was mentioned in introduction. Entire BESIP is freely available under GPL license and binary images from nightly autobild can be downloaded [6].

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 218086.

REFERENCES

[1] Surhone L., Tennoe M., Henssonow M., Susan F., *OpenWRT*, Betascript Publishing, (2011)
[2] Macura L., Voznak M., Tomala K., Slachta J., Embedded Multiplatform SIP Server Solution, In *Proc. 35th International Conference on Telecommunication and Signal Processing*, Prague (2012), 263-266
[3] Macura L., Voznak M., Kamailio syntax generator and configuration file parser, *In Proc. 15th WSEAS International Conference on Computers*, Corfu (2011), 308-312
[4] Enns R. et al., Network Configuration Protocol (NETCONF), *IETF RFC 6241* (2011)
[5] Voznak M., Safarik, J., DoS attacks targeting SIP server and improvements of robustness, *International Journal of Mathematics and Computers in Simulation*, Volume 6 (2012), Issue 1, 177-184
[6] Source code of BESIP Project, LipTel Team, 2011. Available: http://liptel.vsb.cz/mirror/besip/nightly
[7] Cutuli G., Mumolo E., Tessarotto M.,An XML-based virtual machine for distributed computing in a For/Join framework, In *Proc. 24th Int. Conf. Information Technology Interface*, Cavtat, Croatia (2002), 471-477
[8] Chisholm S., Trevino H.,*NETCONF Event Notifications*, IETF RFC 5277 (2008)
[9] Scott M., Bjorklund M., *YANG Module for NETCONF Monitoring*, IETF RFC 6022 (2010)
[10] Libnetconf, NETCONF library in C. Available: http://code.google.com/p/libnetconf
[11] Voznak M., Advanced implementation of IP telephony at Czech universities, WSEAS Transactions on Communications, Volume 9 (2010), Issue 10, 679-693
[12] Voznak M., E-model modification for case of cascade codecs arrangement, *International Journal of Mathematical Models and Methods in Applied Sciences*, Volume 5 (2011), Issue 8, 1439-1447
[13] Transmission impairments due to speech processing, ITU-T Recommendation G.113, Geneva (2007)
[14] Cole G., Rosenbluth, Voice over IP performance monitoring, *ACM SIGCOMM Computer Communication*, New York (2001)
[15] The E-model: A computational model for use in transmission planning, ITU-T Recommendation G.107, Geneva (2011)

***Authors**: Assoc. prof. Miroslav Voznak, Ph.D., MSc. Karel Tomala, MSc. Jiri Vychodil, Jiri Slachta, Technical University of Ostrava, Department of Telecommunications, 17. listopadu 15/2172, 708 33 Ostrava-Poruba, Czech Republic, E-mail: voznak@ieee.org, karel.tomala@vsb.cz, jiri.vychodil@vsb.cz, jiri.slachta@gmail.com.*