Implementing an IP Telephone Exchange Using Asterisk

Best Practice Document

Produced by the AMRES-led working group on Multimedia – VoIP

Author: Ognjen Milosavljevic (RCUB)

April 2016

Document No: GN4-1-NA3-T2-AMRES-BDP-115
Version / date: V1.2 / 20-04-2016
Original language : Serbian
Original title: "*Implementacija IP telefonske centrale korišćenjem besplatnog softverskog paketa Asterisk*"
Original version / date: Version 1 / 4 February 2015
Contact: ognjen.milosavljevic@rcub.bg.ac.rs

AMRES/RCUB is responsible for the contents of this document. The document was developed by the AMRES Multimedia – VoIP (AMRES BPD 115 topic) working group.

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 691567 (GN4-1).

Table of Contents

Table of Figures

Table of Tables

Executive Summary

This paper describes the implementation of an IP telephone exchange at the central location (the headquarters - HQ) of the Academic Network of Serbia (AMRES) using the free Asterisk software package. The paper contains all the instructions necessary for installing and configuring the Asterisk software package, as well as basic information on the accompanying services. The communication of the Asterisk IP Telephone Exchange outside the IP domain is carried out via a SIP (Session Initiation Protocol) trunk provider, and the procedure for its configuration is described in detail. The process of configuring the configuration files necessary for the proper operation of the Asterisk Server is also explained.

Asterisk is the most commonly used open-source telephone communications solution in the IP domain, which uses the SIP protocol. Relying on the IP domain for telephony considerably reduces the costs of communication and simplifies the management of telephone communications.

The first chapter of this paper describes the implementation architecture of the Asterisk IP telephony exchange and the server configuration prior to installing the Asterisk software package. The installation of the Asterisk software package is described in the next chapter, which is followed by a chapter explaining the configuration of the Asterisk IP telephone exchange using the example of the central location of the Academic Network of Serbia. The last two chapters deal with the installation and configuration of the accompanying services, and the configuration of the Iptables tool and SELinux, respectively.

Summary (in Serbian)

U ovom dokumentu je opisana realizacija IP telefonske centrale na centralnoj lokaciji (head quarters - HQ) Akademske mreže Srbije (AMRES), korišćenjem besplatnog softverskog paketa Asterisk. Dokument sadrži sva neophodna uputstva za instalaciju i konfiguraciju Asterisk softverskog paketa, i osnovne informacije o pratećim servisima. Komunikacija Asterisk IP telfonske centrale izvan IP domena realizovana je preko SIP trunk provajdera, i detaljno je objašnjen postupak konfiguracije. Takođe je objašnjeno konfigurisanje konfiguracionih fajlova neophodnih za ispravan rad Asterisk servera.

1 Implementation Architecture

Upon moving to the new location, the AMRES management (HQ) identified the need for a telephony service to be used by its employees. The technical requirements defined that it was necessary to have 6 trunk lines towards the PSTN for communication with public telephone networks, and free SIP/IP communication with the remote service centre of the RCUB (University of Belgrade Computer Centre).

Since the new location did not have an existing telephone network, only the IP domain was selected for the communication between employees, i.e. IP telephony. The Asterisk IP telephone exchange was selected as the solution and connected to a SIP trunk provider for the purposes of communicating with the PSTN network, while a SIP trunk connection has been established with the RCUB through the RCUB Asterisk server by way of the AMRES IP network in order to ensure free and unlimited telephone communication. With the aim of providing a clear picture of the configuration, we will first describe the implementation architecture of the Asterisk IP telephone exchange, as shown in Figure 1.1. The connection with the LAN network of the AMRES HQ is needed in order to enable the registration of and communication between the VoIP (Voice over Internet Protocol) telephones of the AMRES HQ, as well as the communication with other networks. The connection with the SIP trunk provider is necessary to enable calls to the PSTN (public switched telephone network). Finally, the connection with the RCUB Asterisk server (RCUB SIP trunk) is required in order to enable relatively frequent calls between the two locations, which are already connected through the IP network, to be established through the SIP protocol and thus reduce telephone expenses.

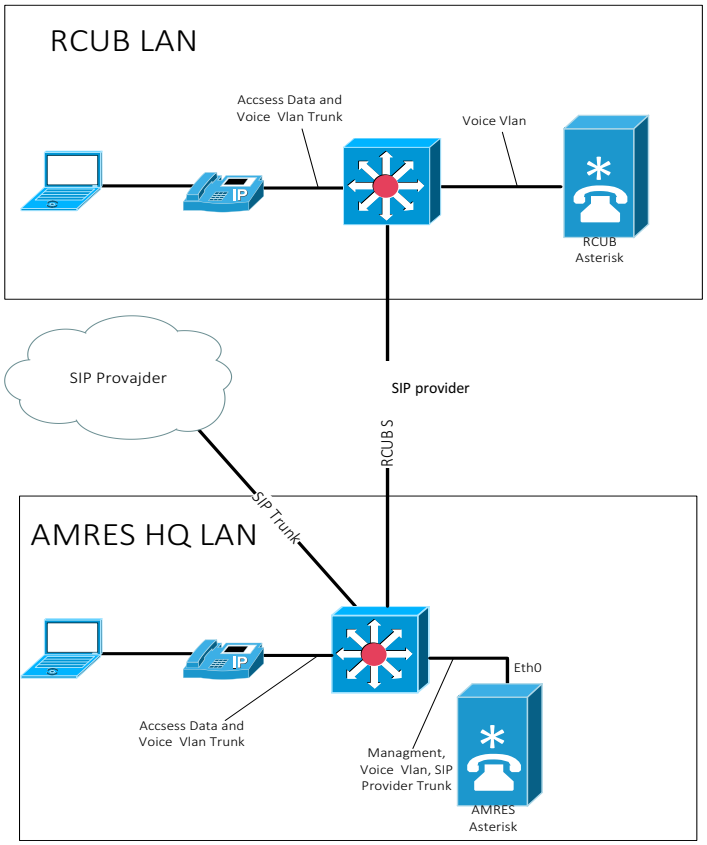

Figure 1.1: The network implementation architecture of the Asterisk IP telephone exchange.

1.1 Configuring the Ethernet card of the AMRES HQ Asterisk server

Since it is necessary for the Asterisk server to communicate with several separate networks, two of which are private (the IP telephony network and the SIP provider network), it is necessary to configure 802.1q tagging on the network card. A public address of the server is configured, which will from now on belong to the untagged VLAN.

It is necessary to configure a private address range dedicated to the VoIP telephones, which will be assigned addresses from that pool via the DHCP protocol. It is also necessary to configure another private address range dedicated to the SIP trunk connection to the SIP provider. The traffic in the VLANs of the VoIP telephones and the communication with the SIP provider needs to be tagged.

After configuring the necessary interfaces, you should check the server routing table and ensure the server has routes for all locally configured networks.

1.1.1 Configuring the hosts and rc.local files for SIP trunk connection

If the SIP trunk is established with a SIP provider's server, it is necessary to configure the *host* and *rc.local* files in the following manner:

- An entry should be made in the *hosts* file so that the domain *example.telecom.com* could be translated into the corresponding IP address. For instance, if the IP address of the SIP provider's server is 10.0.0.2, and its DNS name is *example.telecom.com*, then the following line needs to be entered in the *hosts* file:

```
10.0.0.2 example.telecom.com
```

- It is necessary to configure a static route for the 10.0.0.2 address because the *example.telecom.com* domain is located at this address. One of the ways to configure the static route is to type the following command in the *rc.local* file:

```
route add -net 10.0.0.2 netmask 255.255.255.255 gw 10.0.0.1
```

The example below shows the routing table of the AMRES HQ Asterisk server following the configuration of the *hosts* and *rc.local* files:

```
Destination Gateway Genmask Flags Metric Ref Use Iface
example.telecom.10.0.0.2255.255.255.255 UGH 0 0 0 eth0.251
10.0.0.1* 255.255.255.252 U 0 0 0 eth0.251
203.0.255.0 * 255.255.255.128 U 0 0 0 eth0
10.4.1.0 * 255.255.255.0 U 0 0 0 eth0.254
169.254.0.0 * 255.255.0.0 U 0 0 0 eth0.254
default juniper2200-Amr 0.0.0.0 UG 0 0 0 eth0
```

2 Installing Asterisk software

Prior to the installation of the Asterisk software package, it is necessary to install a Linux operating system. The system selected here is CentOS version 6, and all the commands described herein are applicable to that system.

The installation of the Asterisk software package presented in this paper relies on the Yum package manager. Bearing in mind that the Asterisk software package is not included in the CentOS base repository, it is necessary to create a centos-asterisk.repo file in the /etc/yum.repos.d directory.

The centos-asterisk.repo file should be configured as follows:

```
[asterisk-tested]
name=CentOS-$releasever - Asterisk - Tested
baseurl=http://packages.asterisk.org/centos/$releasever/tested/$basearch
/
enabled=0
gpgcheck=1

[asterisk-current]
name=CentOS-$releasever - Asterisk - Current
baseurl=http://packages.asterisk.org/centos/$releasever/current/$basearc
h/
enabled=1
gpgcheck=1
```

Since the gpg installation check is active, it is necessary to enter a public key into the system before running the installation, which is done using the following command:

```
[root@asterisk ~]# rpm --import
http://packages.asterisk.org/keys/175E41DF.pub
```

The installation process is started using the following command:

```
[root@asterisk ~]# yum install asterisk asterisk-configs asterisk-
voicemail libpri
```

Once the software is installed, it is necessary to reboot the operating system using the following command:

```
[root@asterisk ~]# reboot
```

The following command needs to be entered in order to initiate the Asterisk service:

```
[root@asterisk ~]# service asterisk start
```

Stopping the Asterisk processes requires the following command:

```
[root@asterisk ~]# service asterisk stop
```

To enable the Asterisk software package to start automatically when the system boots, enter the following command:

```
[root@asterisk ~]# chkconfig asterisk on
```

Accessing the Asterisk console line requires the following command:

```
[root@asterisk ~]# asterisk -vvvvvr
Asterisk 1.8.7.0, Copyright (C) 1999 - 2011 Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty'
for details.
This is free software, with components licensed under the GNU General
Public
License version 2 and other licenses; you are welcome to redistribute it
under
certain conditions. Type 'core show license' for details.
=========================================================================
=
 == Parsing '/etc/asterisk/asterisk.conf': == Found
 == Parsing '/etc/asterisk/extconfig.conf': == Found
Connected to Asterisk 1.8.7.0 currently running on asterisk (pid = 2216)
Verbosity was 3 and is now 5
asterisk*CLI>
```

The –r option enables connecting to the Asterisk console line. The amount of detail (verbosity) in the Asterisk console line is defined by the –v option. The more times the –v option is entered when calling the Asterisk command, the more details will be displayed. The increased levels of verbosity of the console line are used for detecting and eliminating problems in the operation of the Asterisk software package.

The following command is used for exiting the Asterisk console line without stopping the Asterisk processes:

```
asterisk*CLI> exit
```

3 Configuring the Asterisk server

The Asterisk server is configured by modifying the configuration files. The two most important configuration files for this purpose are *sip.conf*, which is used to configure the SIP accounts, and *extensions.conf*, which serves for defining the dial plan. It is assumed in this paper that the location of the working folder is /etc/asterisk.

The owner of the configuration files should be the Asterisk user. If this is not the case, change the ownership using the following command:

```
[root@asterisk asterisk]# chown asterisk.asterisk sip.conf

[root@asterisk asterisk]# chown asterisk.asterisk extensions.conf
```

After the installation and before making changes, these files contain detailed explanations of all the configuration lines and their options in the form of comments.

3.1 Configuring the channel file

The *sip.conf* file is the channel configuration file. A channel is the connection by which a call is established with the Asterisk IP exchange. A channel could be a connection leading to a regular telephony device or telephone line, or to a softphone client. The *sip.conf* file contains the configuration for the channel driver, such as *shan_sip.so*, along with the information and credentials required for establishing the connection between the Asterisk IP exchange and a telephony device.

The configuration file is divided into sections, which are referred to as contexts. Contexts are the basic organisational units of a dialplan, and as such they keep different parts of the dialplan independent from each other. The name of a context is written in square brackets and may contain uppercase and lowercase letters, digits (0–9), dash and underscore.

Common information on the channel driver is located at the beginning of the configuration file, more precisely in the *general* context, which all *sip.conf* files begin with. In order for a device (with a suitable account) to be registered with Asterisk, it must have a corresponding context in which the specific parameters for that device/account are defined. The *general* context may also contain information for defining the general configuration parameters for devices (accounts) whose values will be ignored if the parameters are defined in the *account* context or *template* context. The *template* context is used when a large number of contexts have several items in common.

If the values of parameters are not defined in the *account*, *template* or *general* contexts, Asterisk will assign them default values.

Below are examples and descriptions of the contexts in the *sip.conf* file of the AMRES HQ Asterisk IP exchange. In certain cases when describing a context, it will be necessary to refer to another context defined in the *extensions.conf* file.

Parameter	Description
`[general]`	The beginning of the standard *general* context.
`srvlookup=yes`	Enables calling the username@domain type.
`tcpenable=yes`	Enables the server to accept client TCP connections.
`realm=amres.ac.rs`	Used during authentication.
`register=>username@example.telecom.com:password:username@example.telecom.com@example.telecom.com`	Used for registering with the SIP provider example.telecom.com.
`allowguest=yes`	Allows anyone to place a call to the server, including the users behind the NAT device.
`disallow=all`	For preventing the use of codecs; all codecs should be disallowed and then allowed individually.

`allow=alaw`
`allow=gsm`
`allow=ulaw`
`allow=h261`
`allow=h263`
`allow=h263p`
`allow=h264`
	For turning on codecs in a desired order.

`videosupport=yes`	Allows SIP video during communication.
`qualify=yes`	Asterisk checks whether a SIP account is active and waits for a response for a specified time. If it does not receive an answer, it assumes the account is not available for future calls. It is also used for enabling open UDP sessions for SIP accounts located behind a NAT device.
`notifyringing=yes`	Enables notifications when the account is receiving a call.
`allowsubscribe=yes`	Monitoring of the account status is enabled.
`notifyhold=yes`	Monitoring of the call waiting status is enabled.
`callcounter=yes`	Enables Asterisk to receive information on the status of a SIP account.
`dtmfmode=info`	Enables signalling via VoIP telephone during the call.

Table 3.1: Configuring the general context

Table 3.2 below shows the context for configuring the standard SIP account channel:

Parameter	Description
`[petar.petrovic]`	Defining the context of the SIP account channel.
`type=friend`	The *friend* value of the *type* parameter enables accepting incoming and outgoing calls; Asterisk checks the usernames for incoming calls.
`username=petar.petrovic`	Username of the SIP account, used when registering the SIP account.
`secret=******`	User password, used when registering the SIP account.
`qualify=no`	The availability of the SIP client is not checked.
`host=dynamic`	The parameters of the SIP client, hostname or IP address are dynamic and it is necessary for the client to register so that the server can know these parameters.
`canreinvite=no`	If the *canreinvite* parameter is configured as *no*, the traffic will not go directly between SIP clients, but between the Asterisk server and the SIP client.
`context=korisnik`	Defines the usage of the *user* context, which is configured in the *extensions.config* file specifying which SIP account can place calls.
`callerid="Petar Petrovic" <100>`	Defines how the SIP account will identify itself when placing a call.
`subscribecontext=hint-blf`	Defines the specific context for monitoring the status of SIP accounts.
`call-limit=10`	Defines the total number of simultaneous calls.
`callgroup=1`	Defines the group to which the SIP account belongs.
`pickupgroup=1`	Defines the group in which the SIP account can take over calls using the *8 button combination.

Table 3.2: Configuring the standard SIP account channel

Table 3.3 shows the context for configuring the mobile SIP account channel:

Parameter	Description
`[petar.mob]`	Defines the context of the SIP account channel.
`type=friend`	The *friend* value of the *type* parameter enables accepting incoming and outgoing calls; Asterisk checks the usernames for incoming calls.
`username=patar.mob`	Username of the SIP account, used when registering the SIP account.
`secret=******`	User password, used when registering the SIP account.
`qualify=yes`	The availability of the client is checked.
`nat=yes`	Enables SIP clients located behind a NAT device to register with the server.
`host=dynamic`	The parameters of the SIP client, hostname or IP address are dynamic and it is necessary for the client to register so that the server can know these parameters.
`canreinvite=no`	If the *canreinvite* parameter is configured as *no*, the traffic will not go directly between SIP clients, but between the Asterisk server and the SIP client.

`allow=ilbc`
`allow=gsm`
`allow=ulaw`
`allow=alaw`
	Besides the default codecs, additional ones are defined.

`context=korisnik`	Defines the usage of the *user* context, which is configured in the *extensions.config* file specifying which SIP account can place calls.
`callerid=" Petar Petrovic " <300>`	Defines how the SIP account will identify itself when placing a call.

Table 3.3: Configuring the mobile SIP account channel

Table 3.4 shows the configuration of the SIP trunk towards the RCUB Asterisk sever:

Parameter	Description
`[rcub]`	Defines the context of the SIP account channel.
`type=friend`	The *friend* value of the *type* parameter enables accepting incoming and outgoing calls; Asterisk checks the usernames for incoming calls.
`host=203.0.113.226`	Defines the address where the Asterisk server with which the trunk connection is established is located.
`qualify=yes`	The availability of the server is checked.
`disallow=all` `allow=alaw,ulaw`	All codecs should be disallowed and then allowed individually.
`dtmfmode=rfc2833`	Enables DTMF signalling via VoIP telephone during the call.
`insecure=port,invite`	Authentication of an incoming INVITE message is not required in order to start a session, and the port on which the registration request arrives is ignored.
`canreinvite=no`	If the *canreinvite* parameter is configured as *no*, the traffic will not go directly between SIP clients, but between the Asterisk server and the SIP client.
`context=rcub_incoming`	The *rcub* context is associated with the defined *rcub_incoming* context in the *extensions.conf* file.

Table 3.4: Configuring the SIP trunk towards the RCUB Asterisk sever

Table 3.5 shows the configuration of the SIP trunk towards the server of the SIP provider:

Parameter	Description
`[Telekom]`	Defines the context of the SIP trunk towards the SIP provider.
`type=peer`	Defining the channel of the SIP trunk to which calls are placed; default configuration for the SIP provider.
`host=example.telecom.com`	Defines the URL of the server the Asterisk server needs to register with in order to establish the SIP trunk connection.
`disallow=all` `allow=alaw,ulaw`	All codecs should be disallowed and then allowed individually.
`dtmfmode=rfc2833`	Enables DTMF signalling via VoIP telephone during the call.
`insecure=port,invite`	Authentication of the incoming INVITE message is not required in order to start a session, and the port on which the registration request arrives is ignored.
`canreinvite=no`	If the *canreinvite* parameter is configured as *no*, the traffic will not go directly between SIP clients, but between the Asterisk server and the SIP client.
`context=get-telekom-did`	The [Telekom] context is associated with the *get-telekom-did* context defined in the *extensions.conf* file.
`fromdomain=example.telecom.com`	Defining the domain of the SIP provider.
`port=5060`	Defining the port on the SIP trunk.
`secret=******`	Defining the password on the SIP trunk towards the SIP provider.
`username=381117158935`	Defining the username on the SIP trunk towards the SIP provider.

Table 3.5: Configuring the SIP trunk towards the server of the SIP provider

3.2 Defining the dialplan

The dialplan defines how incoming and outbound calls are handled on the Asterisk server. It is defined in the *extensions.conf* file and it comprises the most important part of the Asterisk server configuration. The dialplan is a form of script language containing instructions to be executed by Asterisk when prompted from the outside. (e.g. by a call).

The dialplan is divided into various sections, which are called contexts. Contexts are basic organisational units within the dialplan and as such they keep different parts of the dialplan independent from each other. The name of a context is written in square brackets and may contain uppercase and lowercase letters, digits (0–9), dash and underscore. Within each context, one or more extensions can be defined. An extension is a named sets of actions. An extension defined in one context is isolated from the extension defined in another context, unless their interaction is specified within the instructions of the relevant extension.

Every *extension.conf* file should begin with the *general* context, which contains the general settings of the dialplan. In the case of the AMRES HQ Asterisk server, there is only one parameter defined there and it has the following default value:

```
[general]
autofallthrough=yes
```

If autofallthrough=yes is set, then after the last task is performed within the extension, the call will be terminated with a BUSY signal, CONGESTION signal or HANGUP signal, depending on what Asterisk deems best. If autofallthrough=no is set, then after the extension has performed the last task within the extension, Asterisk will wait for a new extension to be dialled, which is not recommended.

3.2.1 Extensions

Extensions are sequences of actions (where each step contains an application) by which Asterisk conducts a call through the system. Within each context, we can configure as many extensions as necessary. When an extension is prompted by a call, Asterisk will follow the steps defined for the given account. Extensions have the following attributes: name, priority and application.

```
exten =>name,priority,application()
```

The name of an extension can be any combination of letters and numbers, i.e. it does not have to be viewed only as a number being dialled. Every extension has a sequence of steps and every step is identified by its priority.

Steps are arranged in a sequence of numbers, starting with number 1. The *answer* application, which is given number 1, will be executed first and this will be followed by the other applications in order of priority, ending with the *hangup()* application under number 5.

```
exten => 100,1,Answer()
exten => 100,2,aplikacija-2
```

```
exten => 100,3,aplikacija-3
exten => 100,4,aplikacija-4
exten => 100,5,Hangup()
```

If it is necessary to add a new application whose order of execution should be within the existing list of applications, it is necessary to change the order of priority of the applications listed below the inserted one. This method might be rather unscalable and subject to errors. It is for these reasons that the designation "n" (next) is introduced when assigning priority numbers to applications. The application with the priority number "n" will thus take over the value of the priority number of the preceding application in the sequence, increased by one.

```
exten => 100,1,Answer()
exten => 100,n,aplikacija-2
exten => 100,n,aplikacija-3
exten => 100,n,aplikacija-4
exten => 100,n,Hangup()
```

Application is a command that is executed during a certain step of the extension. Applications usually require an argument defined in brackets in order to execute their tasks, though there are applications that do not require such arguments.

```
exten => 200,1,application(one,two,three)
```

3.3 The link between the channel and the dialplan

One of the required parameters when configuring the SIP account channel in the *sip.conf* file is the context. Within the *sip.conf* file, the context indicates the position in the dialplan from which the connection for a given channel starts. The link between the *sip.conf* file and the extension.conf file is shown in Figure 3.1

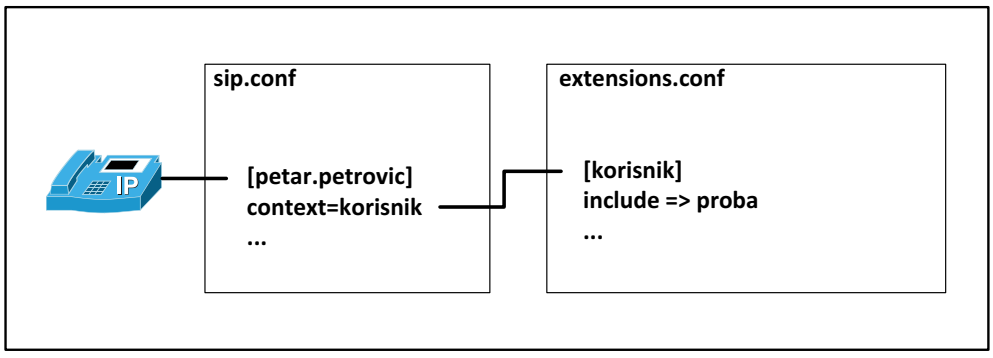

Figure 3.1: The link between the *sip.conf* and *extension.conf* files

Context is very important in terms of the security of Asterisk, since we can use it to control the destinations to which calls can be placed from certain accounts. For instance, defining a separate context for an ordinary user so that it only allows local calls reduces the risk of unauthorised international calls, because such calls are only permitted for users within the advanced users' context.

Given below is a description of the link between the channel configuration (*sip.conf* file) and the dialplan (*extensions.conf* file) shown using the examples of calls between:

- Users registered with the AMRES HQ Asterisk only.
- Users registered with the AMRES HQ Asterisk and users registered with the RCUB Asterisk server.
- Users registered with the AMRES HQ Asterisk and telephones on the public switch telephone network (PSTN), and vice versa.

The description of these links will include the configuration of the used parts of the *extensions.conf* file.

Users of local and mobile accounts who are registered with the AMRES HQ Asterisk server are located within the *user* (*korisnik*) context.

The first example (Figure 3.2) shows a call established between two users registered with the AMRES HQ Asterisk server; more precisely, korisnik1 is calling korisnik2:

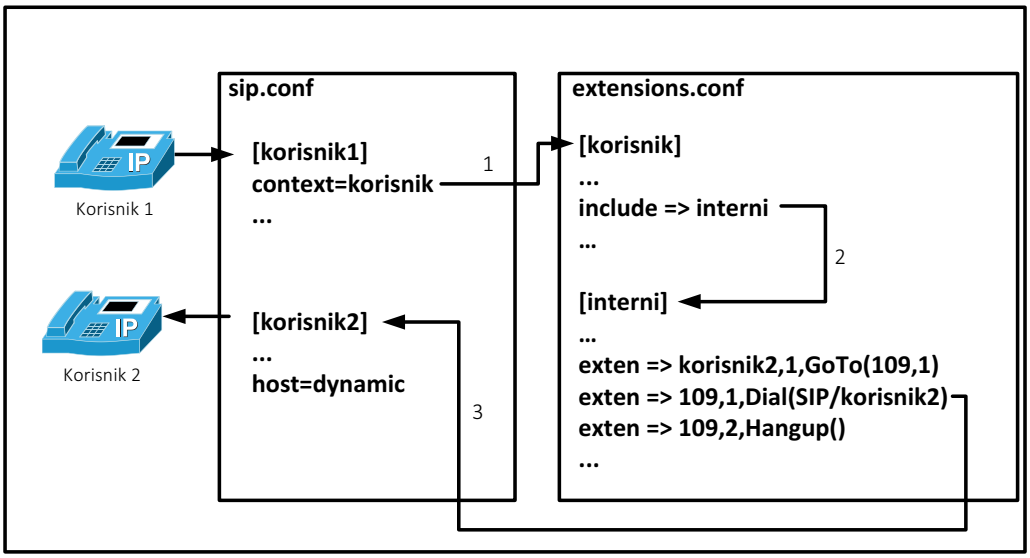

Figure 3.2: Example 1, *sip.conf* and *extensions.conf* files during a call between users on the AMRES HQ Asterisk server

Image 3 (shows that the *user* context includes (the key word being "include") the *internal* (*interni*) context, in which the "109" extension called by *korisnik1* is located. After the number 109 is called, step 1 forwards the call to the *user* context, which further forwards it to the *internal* context in step 2, were the line *exten => 109,1,Dial(SIP/korisnik2)* is located. The *Dial()* application is executed first, together with its additional elements. The *Dial()* application enables the calling of one of the SIP account channels. When configuring the extension and the user *korisnik2*, the first line contains the *GoTo()* application, which enables the call to be redirected elsewhere within the dialplan, in our case to extension 109. The *Hangup()* application is almost always used at the end of the definition of each extension in order to unconditionally terminate the call.

Example 2 (Figure 3.3) shows a call that a user registered with the AMRES HQ Asterisk server establishes with a user registered with the RCUB Asterisk server; more precisely, *korisnik1* is calling *korisnik1rcub* at the number 5122:

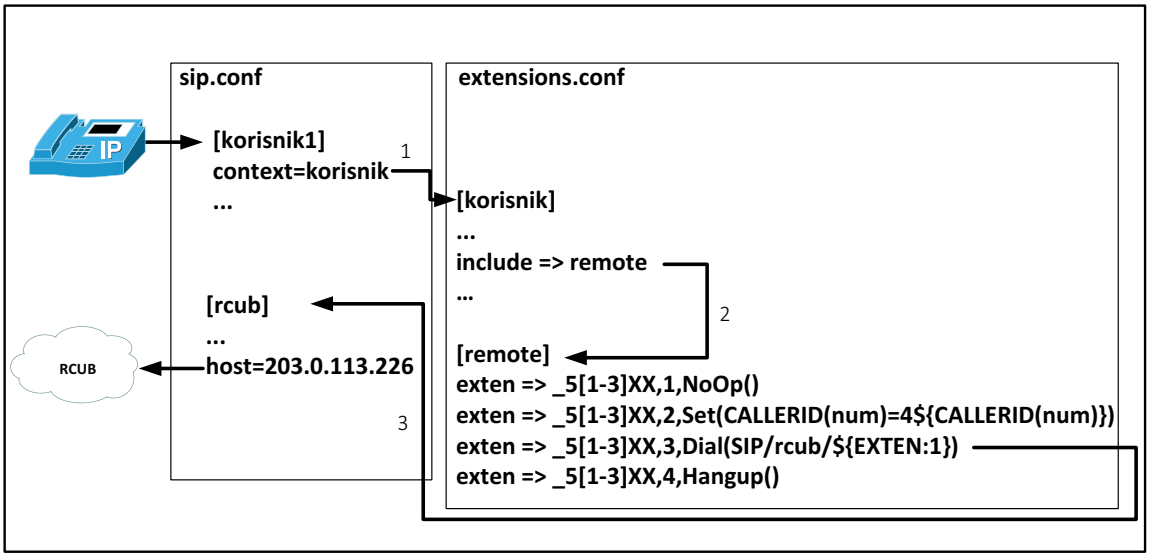

Figure 3.3: Example 2, *sip.conf* and *extensions.conf* files during a call a user registered with the AMRES HQ Asterisk server places to a user registered with the RCUB Asterisk server

In the third example, when the number 5122 is dialled, step 1 forwards the call to the *user* (*korisnik*) context, which in turn forwards it to the *remote* context in step 2. The *remote* context contains the extension that further routes the call to the number 5122, because it includes the line that uses the "_5[1-3]XX" template, which coincides with the four digit number combinations beginning with "5", where the second digit is a number from 1 to 3 and the fourth and fifth digits are numbers from 0 to 9.

The *NoOp()* is an application that can be used to write the values of a variable, provided that the variable is placed within brackets, e.g. NoOp (${CALLERID}). The CALLERID variable shows the value with which the caller is introduced to the called party. It can be in the form of a number, a name or both.

The *Set()* application serves for changing the values of variables, but not all of them; e.g. the ${EXTEN} variable cannot be changed using the *Set()* application. In our example, the value of the CALLERID(num) variable is changed, i.e. we are adding the prefix 4 in front of the numeric value of the caller identification. The purpose of this step is to enable the called user to subsequently search the call history and call the user who called from another Asterisk server.

The ${EXTEN} variable shows the called number. The call needs to be directed to the IP address 203.0.113.226, where the RCUBA Asterisk server is located. This server uses the enumeration from "[1-3]XX" for its users, so it is necessary to remove number "5" so that the RCUB Asterisk can correctly route it further. The removal of the first number is achieved using the ${EXTEN:x} syntax, where x is the number of digits to be removed, counting from left-to-right.

Example 4 (Figure 3.4)shows a call a user registered with the AMRES HQ Asterisk server establishes with a user on the public switch telephone network (PSTN). The user *korisnik1* is calling the number 03020703 (the call is within the same town, so there is no need for an area code), where the digit 0 is dialled for exiting to the PSTN via the SIP trunk provider.

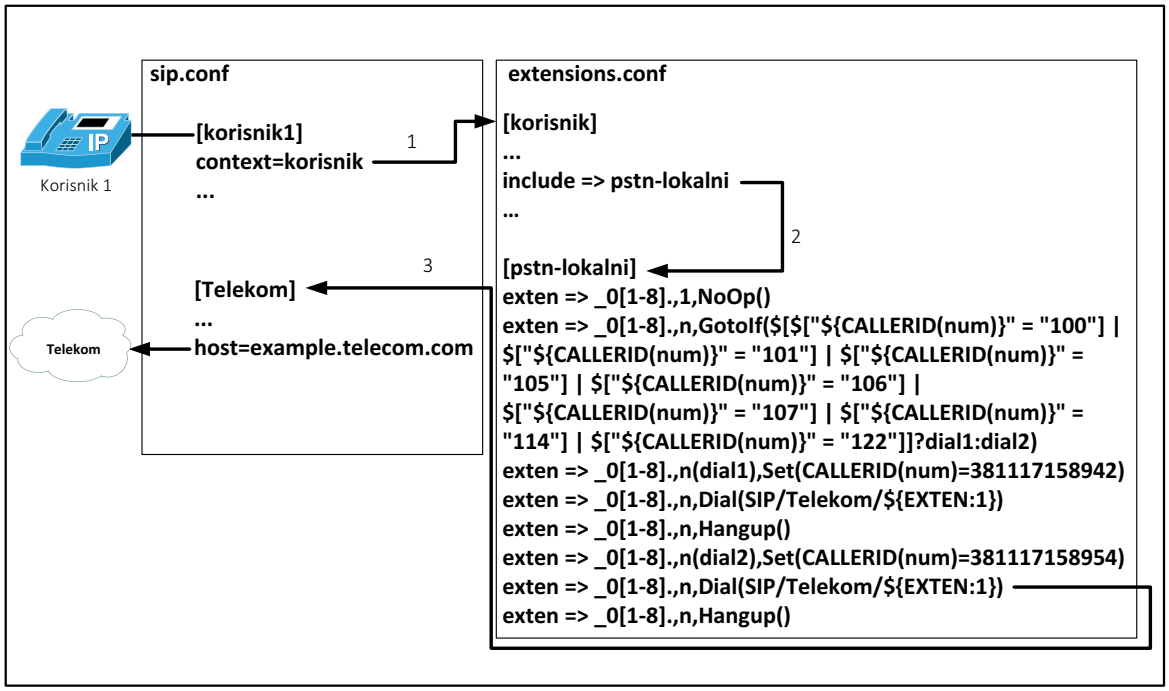

Figure 3.4: Example 4, calling a PSTN number

Upon dialling the number 03020703, step 1 forwards the call to the *user (korisnik)* context, from where it is further forwarded to the *pstn-lokalni* (local) context in step 2, where it reaches the *GoToIf()* application that enables moving to a specific position in the extension, based on the requirement set in the application. Specifically, the requirement *dial1* is applied to the extensions 100, 101, 105, 106, 107, 114 or 122 when calling a number on the PSTN, while the requirement *dial2* is applied to all other extensions. When the *dial1* requirement is met, the *GoToIf()* application forwards the call to the line where all the above users are given the CALLERID "38117158942", and when the *dial2* requirement is met the application forwards the call to the line where all the other users are given CALLERID "38117158954".

After the CALLERID parameter is selected, step 3 forwards the call to the SIP trunk under the *Telekom* context.

Example 5 (Figure 3.5) shows a call established by a PSTN user with a user on the AMRES HQ Asterisk server; more precisely, the number 06687331056 is dialling the number 0119158943, which is the user *korisnik1* registered with the AMRES HQ Asterisk server.

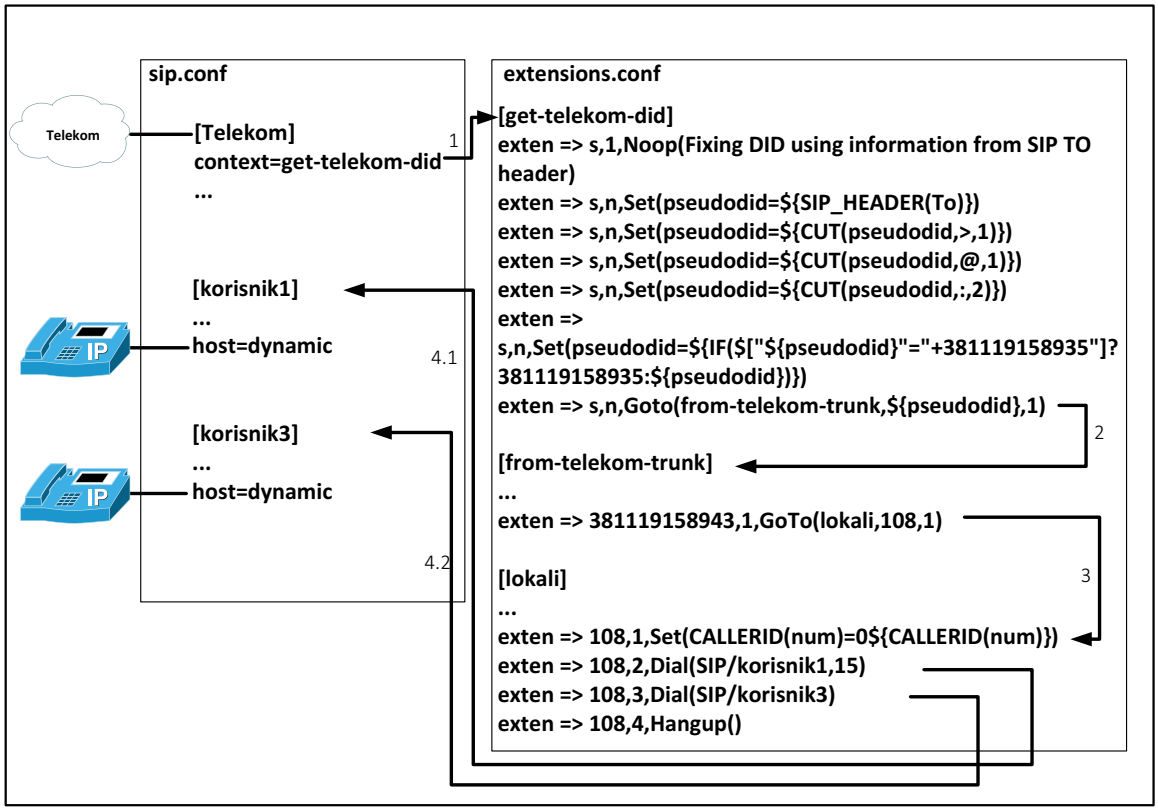

Figure 3.5: Example 5, a call from the PSTN to a user on the AMRES HQ Asterisk server

Before we explain the configuration of example 5, it is necessary to explain what DID is. Telekom has enabled an exit to the PSTN network for up to 6 simultaneous calls. It has also provided 20 telephone numbers i.e. DIDs (Direct Inward Dialling). Certain telephone numbers have been distributed among the users. When someone calls one of the distributed numbers, the call is routed directly to the extension of the user to whom that DID was assigned. Some of the users do not have direct (DID) numbers because there are not enough telephone numbers.

When the number 0119158946 is dialled, step 1 forwards the call to the get-telekom-did context. Since the SIP message coming from Telekom to the AMRES HQ Asterisk server contains information on the incoming number (DID), but not in a form matching the configuration, it is necessary to find the information of interest, more specifically a "pseudodid" with the following form: "381119158XX", where "XX" can have a value ranging from 25 to 44. As seen in image 6, when the information about the "pseudodid" is obtained, step 2 moves us to the new context, from-telekom-trunk, which forwards the call in step 3 to the lokali (extensions) context where the final routing of the call is performed, through step 4.1 or 4.2, to the specific extension connected to the DID being called.

Note: the section entitled "Configuring the Business Trunk of Telekom Srbija" provides more information on the SIP trunk connection, i.e. the Business Trunk of Telekom Srbija.

Special extensions, which have a special purpose in the dialplan, are also configured in the extensions.conf file. They are used for certain situations during the call, as follows:

```
exten => t,1,Hangup
```

The “t” extension is used when it is necessary to define what to do when information is being awaited for a long time. For instance, a user has picked up the handset but has not dialled a number for a long time.

```
exten => i,1,Congestion
```

The “i” extension is used when it is necessary to define what to do when invalid information is received, e.g. when someone dials an extension that is not found in any context. The Congestion() application indicates to the calling party that the line is congested, which is similar to the Busy() application that notifies the calling party that the line is busy.

```
exten => s,1,Congestion
```

The “s” extension is used when it is necessary to define what to do when unexpected information is received. For instance, when a call is being received from an FXO card, no other information is expected than the ringing tone.

```
exten => h,1,Hangup
```

The “h” extension serves for defining what to do when the calling party hangs up. For instance, if the user hangs up during the call, the call is also disconnected on the side of the dialled extension.

3.4 Configuring the SIP trunk connection with the RCUB Asterisk server

When connecting two Asterisk servers through a SIP trunk, it is necessary to define prefixes for each of them so that the local Asterisk server can establish calls with a remote Asterisk server. The selected prefixes should not coincide with the already selected prefixes for exiting to the public switched telephone network (PSTN) or to another telephone network. On the RCUB Asterisk server, the number 4 was selected as the prefix for the AMRES HQ Asterisk server, while on the AMRES HQ Asterisk server the number 5 was selected.

This section describes the configuration of the SIP trunk connection with another Asterisk server located at the IP address 203.0.113.226. It is necessary to configure the *sip.conf* and *extensions.conf* files in the following way:

Below is the part of the *sip.conf* file configuration that is in charge of this:

```
[rcub]
type=friend
host=203.0.113.226
qualify=yes
disallow=all
allow=alaw,ulaw
```

```
dtmfmode=rfc2833
insecure=port,invite
canreinvite=no
context=rcub_incoming
```

The following part of the extensions.conf file is in charge of incoming calls from the RCUB Asterisk server to the AMRES HQ Asterisk server:

```
[rcub_incoming]
include => interni
include => mobilni
```

The part of the *extensions.conf* file configuration responsible for outgoing calls from the AMRES HQ Asterisk server to the RCUB Asterisk server is as follows:

```
[remote]
exten => _5[1-3]XX,1,NoOp()
exten => _5[1-3]XX,2,Set(CALLERID(num)=4${CALLERID(num)})
exten => _5[1-3]XX,3,Dial(SIP/rcub/${EXTEN:1})
exten => _5[1-3]XX,4,Hangup()

[korisnik]
include => remote
```

3.5 Configuring the Telekom Srbija business trunk

The Business Trunk service of Telekom Srbija is used to ensure the exit of VoIP traffic to the PSTN. The Business Trunk service enables the connection of the IP telephone exchange (in our case the Asterisk server), via the provider's IP network, to the SIP platform for connecting to the public switched telephone network (PSTN).

The following information is received from the SIP trunk provider, which enables the provision of the service by the provider. It is necessary to establish suitable SIP signalling between the IP PBX/voice gateway of our device and Telekom's IMS (IP multimedia core network subsystem):

- The AMRES HQ Asterisk server needs to forward the registration request, the *REGISTER* message with its SIP address, with which it is registered to the IMS and which represents all the local users of the IP telephone exchange.
- The *REGISTER* message must contain the following:
  - the request URI header, which contains the home network domain (REGISTER sip:example.telecom.com SIP/2.0)
  - the from and to fields of the message header must contain the primary SIP URI public user identity of the trunk group
  - the AMRES HQ Asterisk server needs to forward the second REGISTER message containing the nonce and response values, as well as the username containing the SIP URI public user identity of the trunk group.

- The AMRES HQ Asterisk server should use the *domain name* as part of the SIP URI within the following fields: *from, to, p-asserted-identity* and *request-URI*, for calls forwarded form the Asterisk server to the IMS. The IP address of the user device is not used in any of these messages.
- For outgoing calls forwarded from the Asterisk server to the IMS, the server sends the INVITE message with the p-asserted-identity (PAI) field, which needs to contain the SIP URI public user identity of the trunk group, based on which the IMS application server identifies the trunk group of the Asterisk server, i.e. the IP of the telephone exchange. The from field of the INVITE message should contain the calling line identity (CLID) of the user on the IP telephone exchange who placed the call, i.e. of the corresponding trunk user. The from field should include the telephone URI or the SIP URI. Based on the content of the from field, the IMS application server identifies the user of the IP telephone exchange that places the call.
- For incoming calls forwarded from the IMS to the Asterisk IP telephone exchange, the local user call is identified by the *to* field of the *INVITE* message received from the IMS. This field contains the primary *SIP URI* identifier of the trunk user the call is placed to. The *request-URI* field in the *INVITE* message includes the primary *SIP URI* public user identity of the trunk group. Based on the *request-URI* field, the IMS application server identifies the corresponding trunk group of the IP telephone exchange and forwards the call to that trunk.

In short, in order to register the IP, the telephone exchange sends the *REGISTER* message by which it is registered with the IMS and which represents all its local users. Outgoing calls are routed based on the contents of the *PAI* and *from* fields of the INVITE message, while incoming calls are routed based on the *to* and *request-URI* fields.

Local traffic originating and ending within the IP telephone exchange does not need to be forwarded to the IMS.

The next thing that needs to be done is to configure the *sip.conf* and *extension.conf* files in the following way. The part of the *sip.conf* file configuration in charge of the configuration of the SIP trunk towards the provider is shown below (the password is marked by *****):

```
[general]
register => 381119158935@example.telecom.com:*****:381119158935@
example.telecom.com@ example.telecom.com

[Telekom]
type=peer
host= example.telecom.com
disallow=all
allow=alaw,ulaw
dtmfmode=rfc2833
insecure=port,invite
canreinvite=no
context=get-telekom-did
fromdomain=example.telecom.com
port=5060
secret=*****
username=381119158935
```

The part of the *extensions.conf* file configuration that is in charge of the configuration of the SIP trunk towards the provider is shown below (this part of the configuration concerns calls coming from the PSTN to the AMRES HQ Asterisk server):

```
[get-telekom-did]
exten => s,1,Noop(Fixing DID using information from SIP TO header)
exten => s,n,Set(pseudodid=${SIP_HEADER(To)})
exten => s,n,Set(pseudodid=${CUT(pseudodid,>,1)})
exten => s,n,Set(pseudodid=${CUT(pseudodid,@,1)})
exten => s,n,Set(pseudodid=${CUT(pseudodid,:,2)})
exten =>
s,n,Set(pseudodid=${IF($["${pseudodid}"="+381119158935"]?381119158935:${
pseudodid})})
exten => s,n,Goto(from-telekom-trunk,${pseudodid},1)

[from-telekom-trunk]
exten => 381119158935,1,GoTo(lokali,100,1)
…

[lokali]
exten => 100,1,Set(CALLERID(num)=0${CALLERID(num)})
exten => 100,2,Dial(SIP/zoran.jovanovic,15)
…
```

The part of the configuration in the *extensions.conf* file responsible for this is shown below (this part of the configuration concerns outgoing calls from the AMRES HQ Asterisk to the PSTN):

```
[pstn-lokalni]
exten => _0[1-8].,1,NoOp()
exten => _0[1-8].,n,GotoIf($[$["${CALLERID(num)}" = "100"] |
$["${CALLERID(num)}" = "101"] | $["${CALLERID(num)}" = "105"] |
$["${CALLERID(num)}" = "106"] | $["${CALLERID(num)}" = "107"] |
$["${CALLERID(num)}" = "114"]]?dial1:dial2)
exten => _0[1-8].,n(dial1),Set(CALLERID(num)=381119158942)
exten => _0[1-8].,n,Dial(SIP/Telekom/${EXTEN:1})
exten => _0[1-8].,n,Hangup()
exten => _0[1-8].,n(dial2),Set(CALLERID(num)=381117158954)
exten => _0[1-8].,n,Dial(SIP/Telekom/${EXTEN:1})
exten => _0[1-8].,n,Hangup()
```

The same configuration applies to the *pstn-međugradski* (domestic long-distance), *pstn-mobilni* (mobile), *pstn-hitni* (emergency) and *pstn-besplatni* (free) contexts, save for the patters that are different for each context.

```
[korisnik]
include => pstn-lokalni
include => pstn-medjugradski
include => pstn-mobilni
include => pstn-medjunarodni
include => pstn-hitni
include => pstn-besplatni
```

3.6 Configuring SIP URI calls

The configuration of SIP URI calls poses a security risk because it enables unregistered users to make calls, while, on the other hand, it enables calls via the URI if necessary.

In order to enable incoming SIP URI calls, it is necessary to make the appropriate SRV entries in the DNS zone table, which is *amres.ac.rs* in our example, and the *asterisk.amres.ac.rs* server is located in that zone. Allowing SIP URI calls requires the *sip.conf* file to be configured accordingly, so that all unauthorised incoming calls are directed to the right context in the *extensions.conf* file.

It is necessary to add the following lines in the zone table:

```
_sip._tcp SRV 0 0 5060 asterisk.amres.ac.rs.
_sip._udp SRV 0 0 5060 asterisk.amres.ac.rs.
_sip._tcp.asterisk SRV 0 0 5060 asterisk.amres.ac.rs.
_sip._udp.asterisk SRV 0 0 5060 asterisk.amres.ac.rs.
```

The default context that is called from the *extensions.conf* file should be defined as *sip-dolazni* (incoming) for all users who want to place calls to the Asterisk server without having it individually configured within their respective channel contexts. The part of the *extensions.conf* file configuration in charge of receiving URI calls is shown below:

```
[sip-dolazni]
include => interni
```

It is necessary to configure the following in the *sip.conf* file, within the *general* context:

```
[general]
context=sip-dolazni
…
```

It is also necessary to allow the unauthorised user *anyone* to establish a SIP URI call, which is done by the following configuration:

```
[anyone]
type=friend
host=dynamic
qualify=yes
disallow=all
allow=alaw,ulaw
dtmfmode=rfc2833
insecure=port,invite
canreinvite=no
```

The call to the SIP URI address of the user can only be forwarded to extensions (local numbers) and no outside calls can be placed through the Asterisk server, which comprises a form of protection.

For the purpose of making outgoing SIP URI calls, it is necessary to configure the following in the *sip.conf* file:

```
[general]
srvlookup=yes
…
```

The part of the *extensions.conf* file configuration in charge of sending the URI is as follows:

```
[korisnik]
include => dial-uri
…
[dial-uri]
exten => _[a-z].,1,Macro(uridial,${EXTEN}@${SIPDOMAIN})
exten => _[A-Z].,1,Macro(uridial,${EXTEN}@${SIPDOMAIN})
exten => _X.,1,Macro(uridial,${EXTEN}@${SIPDOMAIN})
exten => _+.,1,Macro(uridial,${EXTEN}@${SIPDOMAIN})
[macro-uridial]
exten => s,1,Set(dialuri=${CUT(ARG1,\;,1)})
exten =>
s,n,ExecIf($["${DB(${CALLERID(number)}/user_sipname)}"!=""],Set,CALLERID
(number)=${DB(${CALLERID(number)}/user_sipname)})
exten => s,n,NoOp(Calling SIP URI ${dialuri})
exten => s,n,NoOp(--- From: ${CALLERID(all)} ---)
exten => s,n,Dial(SIP/${dialuri},120,tr)
exten => s,n,Congestion()
```

3.7 Configuring the attendant console

Another functionality of the Asterisk system is monitoring the state of extensions. Namely, Asterisk as a system can perform checks in case one of the extensions is busy, unavailable or shows any other status. In order to determine the state of the extension, we use the *hint* directive within the *hint-blf* context in the *extensions.conf* file:

```
[hint-blf]
exten => 100,hint,SIP/korisnik1
…
```

In order to check the status of the extension, it is necessary to run the following command in the Asterisk CLI:

```
asterisk*CLI> core show hints

-= Registered Asterisk     Dial Plan   Hints =-
104@hint-blf : SIP/korisnik1     State:Idle Watchers 5
105@hint-blf : SIP/korisnik2 State:Idle Watchers 5
106@hint-blf : SIP/korisnik3 State:Idle Watchers 5
 .
 .
 .
107@hint-blf : SIP/korisnik30 State:Idle Watchers 5
199@hint-blf : SIP/amreshelpdesk State:Idle Watchers 5
```

The *EXTENSION_STATE()* function serves for checking the status of the given extension. The *EXTENSION_STATE()* function can return the following values:

1. UNKNOWN
2. NOT_INUSE
3. INUSE
4. BUSY
5. UNAVAILABLE
6. RINGING
7. RINGINUSE
8. HOLDINUSE
9. ONHOLD

Asterisk enables some devices, which in our case is the attendant console, to be notified about the state of the extension. Besides the part of the configuration in the *extensions.conf* file, the *sip.conf* file also needs to be configured:

```
[general]

allowsubscribe=yes
notifyringing=yes
limitonpeer=yes
notifyhold=yes
callcounter=yes

[korisnik10]
…
subscribecontext=hint-blf
```

This functionality is usually called BLF ("busy lamp field") because it enables the information about the state of extensions to be identified by the colour of buttons on the console. It is necessary to supplement the .xml files of the VoIP telephone devices the attendant console is connected to, i.e. the part related to the console.

New extensions require the addition of the following line in the .xml files of the VoIP telephone devices connected to the attendant console:

```
<Unit_broj po redu _Key_broj po redu group="Attendant_Console/Unit_broj
po redu">fnc=sd+cp+blf;sub=lokal@
IP_ASTERIKS;nme=lokal;usr=ime.prezime@IP_ASTERIKS
</Unit_broj po redu_Key_broj po redu>
```

Below is an example in which 192.0.2.126 is the IP address of the server:

```
<Unit_1_Key_1
group="Attendant_Console/Unit_1">fnc=sd+cp+blf;sub=100@192.0.2.126;nme=1
00;usr=korisnik10@192.0.2.126
```

```
</Unit_1_Key_1>
```

3.8 The Asterisk command line

The following command is used in the Asterisk CLI for checking the SIP accounts defined in the *sip.conf* file:

```
asterisk*CLI>sip show peers
Name/username Host Dyn Forcerport ACL Port Status
Telekom/381119158935 10.0.0.2          N 5060 OK (27 ms)
amreshelpdesk/amreshelpde 10.4.1.254 N 5060 Unmonitored
korisnik1/korisnik1  10.4.1.108 D      N 5060 Unmonitored
korisnik1.mob/ korisnik( Unspecified) D      N 0 UNKNOWN
korisnik2/ korisnik2 10.4.1.104 D      N 5060 Unmonitored
korisnik2.mob/ korisnik2 (Unspecified) D     N 0 UNKNOWN
.
.
.
rcub 203.0.113.226 N 5060 OK (1 ms)
sipp (Unspecified) N 0 UNKNOWN
.
.
.
39 sip peers [Monitored: 3 online, 12 offline Unmonitored: 24 online, 0
offline]
```

In order to check all the active channels, we use the following command in the Asterisk CLI:

```
asterisk*CLI>sip show channels
Peer User/ANR Call ID Format Hold Last Message Expiry Peer
203.0.113.226 (None) 0a36e9a91c7510f 0x0 (nothing) No Rx: OPTIONS
<guest>
10.4.1.105 dragana.kovac fb8c4576-5ee7b0 0x4 (ulaw) No Rx: ACK
korisnik12
10.0.0.2 2455062 0d84ea2a6ffafa7 0x8 (alaw) No Tx: ACK Telekom
3 active SIP dialogs
```

The following command in the Asterisk CLI serves for reloading the configuration, e.g. after entering changes:

```
asterisk*CLI>reload
```

Note: the other Asterisk CLI commands can be found at [AsteriskCLI].

3.9 Configuring the ASTDB file

After making the changes on the Asterisk server, it is necessary to restart the entire server, which we do in the following way in the Asterisk mode:

```
asterisk*CLI>core restart now
asterisk*CLI>
Disconnected from Asterisk server
Executing last minute cleanups
[root@asterisk ~]#
```

After the restart, it is necessary to change the permissions for the *astdb* file, where the data on previously registered SIP accounts is kept, because permission is denied to the asterisk user.

By changing the ownership of the file to Asterisk, we have enabled the Asterisk process to read the *astdb* file on every restart of the server and register all the users who were registered prior to switching the server off.

The following command in the Asterisk CLI is used for checking the defined SIP accounts in the *sip.conf* file:

```
asterisk*CLI>sip show peers
Name/username Host Dyn Forcerport ACL Port Status
Telekom/381119158935 10.0.0.2          N 5060 OK (27 ms)
amreshelpdesk/amreshelpde 10.4.1.254 N 5060 Unmonitored
korisnik1/korisnik1  10.4.1.108 D      N 5060 Unmonitored
korisnik1.mob/ korisnik( Unspecified) D      N 0 UNKNOWN
korisnik2/ korisnik2 10.4.1.104 D      N 5060 Unmonitored
korisnik2.mob/ korisnik2 (Unspecified) D     N 0 UNKNOWN
.
.
.
rcub 203.0.113.226 N 5060 OK (1 ms)
sipp (Unspecified) N 0 UNKNOWN
.
.
.
39 sip peers [Monitored: 3 online, 12 offline Unmonitored: 24 online, 0
offline]
```

In order to check all the active channels, we need to enter the following command in the Asterisk CLI:

```
asterisk*CLI>sip show channels
Peer User/ANR Call ID Format Hold Last Message Expiry Peer
203.0.113.226 (None) 0a36e9a91c7510f 0x0 (nothing) No Rx: OPTIONS
<guest>
10.4.1.105 dragana.kovac fb8c4576-5ee7b0 0x4 (ulaw) No Rx: ACK
korisnik12
10.0.0.2 2455062 0d84ea2a6ffafa7 0x8 (alaw) No Tx: ACK Telekom
3 active SIP dialogs
```

Note that the other Asterisk CLI commands can be found at [AsteriskCLI].

3.10 Viewing logs

The following command enables the viewing of logs in real time:

```
[root@asterisk ~]# less /var/log/asterisk/messages
```

In order to view old logs, it is necessary to enter the following command:

```
[root@asterisk ~]# joe /var/log/asterisk/messages
```

The extent of detail of the messages in the */var/log/asterisk/messages* files is defined within the */etc/asterisk/logger.conf* file. The following command in the *logger.conf* file defines the most detailed logging of system messages:

```
messages => security,notice,warning,error,debug,verbose,dtmf,fax
```

In order to view call logs, it is necessary to enter the following command:

```
[root@asterisk ~]# cd /var/log/asterisk/cdr-csv
[root@asterisk cdr-csv]# joe Master.csv
```

4 Installing and configuring accompanying services

The registration of a VoIP device (hardware or software) requires obtaining information first from the DHCP server, and then from the TFTP server. During this process, the VoIP device receives an address from the DHCP range (or more precisely, the address intended specifically for its MAC address in the *dhcpd.conf* file). The VoIP device will also receive information about the IP address of the TFTP server from the DHCP server, after which it addresses the TFTP server in order to obtain the appropriate.xml configuration file.

The remaining part of this paper will briefly present the installation and configuration of the accompanying services necessary for the automatic provisioning of the client devices in the Asterisk IP telephone exchange.

4.1 Installing and configuring the DHCP server

Within the AMRES HQ VoIP solution, the DHCP server is implemented on the same server as the Asterisk server. The DHCP server can be implemented separately from the Asterisk server, but it then needs to be in the same VLAN as the IP telephones.

The DHCP server can be installed via the *yum* package manager:

```
[root@asterisk ~]# yum install dhcp
```

The DHCP service needs to be configured to run together with the launch of the server:

```
[root@asterisk ~]# chkconfig dhcpd on
```

The following command can be used to check the status and management of the DHCP process:

```
[root@asterisk ~]# service dhcpd status/start/stop/restart
```

The DHCP server configuration file is *dhcpd.conf*. In this file, it is necessary to configure the range of IP addresses that will be assigned to DHCP clients via the DHCP protocol. To configure the DHCP range, it is necessary to define the following line in the *dhcp.conf* file (in the example below, the clients are assigned addresses from the 10.0.20.0/24 range):

```
subnet 10.0.20.0 netmask 255.255.255.0 { }
```

Next, the configuration that assigns an IP address to an IP telephone based on the MAC address is as follows:

```
host user-phone {
 hardware ethernet 00:0b:be:b2:5f:5d;
 fixed-address 10.0.20.231;
 }
```

After the changes have been made to the configuration file of the DHCP server, the DHCP service needs to be restarted.

```
[root@asterisk etc]# service dhcpd start
Starting dhcpd: [ OK ]
```

After all the changes have been made, the service will be launched successfully.

4.2 Installing the TFTP server

The TFTP server is necessary for configuring the VoIP device. To install the TFTP server, the following command needs to be entered:

```
[root@asterisk ~]# yum install tftp-server
```

To launch the TFTP process automatically, it is necessary to enter the following command:

```
[root@asterisk ~]# chkconfig tftp on
```

The TFTP server is started with the help of the Xinetd service, which needs to be installed using the following command:

```
[root@asterisk ~]# yum install xinetd
```

In order to launch the Xinetd process automatically, the following command needs to be entered:

```
[root@asterisk ~]# chkconfig xinetd on
```

The non-automatic launch of the TFTP process will require the following command:

```
[root@asterisk ~]# service xinetd start
```

The purpose of the TFTP server is to ensure configuration files when running VoIP telephones. When a VoIP telephone gets an IP address from the DHCP server, which is predefined based on its MAC address, it also needs to get the address of the TFTP server from the DHCP server. The TFTP server contains the configuration .xml files for VoIP telephones. These files need to be named as follows: SEPMACADRESA.cnf.xml. For instance, the name of the configuration file for the following MAC address: 00:2b:9f:ac:d6:d9 is: SEP002B9FACD6D9.cnf.xml. The content of the .xml files varies depending on the manufacturer and model of the VoIP telephones.

5 Configuring IPtables and SELinux tools

The Asterisk server requires protection to prevent unauthorised access, but at the same time enable communication with clients and other Asterisk servers in order to ensure the functioning of the VoIP service. IPtables tools are used to filter the traffic on the server. The filtering is performed based on source and destination addresses and TCP/UDP ports.

Since the SIP protocol uses TCP 5060, TCP 5061, UDP 5060 and UDP 5061 ports for exchanging signalling messages, it is necessary for the Asterisk server to have access to these ports.

At the AMRES HQ, as a result of the need to register mobile VoIP telephones from external networks (networks that are not local), addresses are not filtered by the SIP ports.

If the incoming traffic to the SIP ports is not filtered, it is necessary to install the *fail2ban* tool in order to prevent brute force attacks on the SIP ports. If there is no need for the registration of VoIP telephones from external networks, it is only necessary to allow access to the IP addresses of the local network in which the VoIP telephones are located and the IP addresses of the SIP trunk server.

For the exchange of media streams, it is necessary to allow access to ports used by the RTP protocol and to UDP ports from 10000 to 20000.

It is necessary to allow access to the port used by the DHCP protocol, the UDP port 67, the port used by the TFTP protocol, the UDP port 69, the port used by the SSH protocol, and the TCP port 22.

Following the modification of the IPtables tool file, it is necessary to reset the IPtables service in order for the changes to take effect.

If the server administrator does not possess any experience in handling the SELinux tool, it is recommended that this tool is disabled since it can limit the privileges of other processes on the server and thus prevent the proper operation of the Asterisk server.

The SElinux tool is disabled in the /etc/selinux/config file:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - SELinux is fully disabled.
SELINUX=disabled
# SELINUXTYPE= type of policy in use. Possible values are:
# targeted - Only targeted network daemons are protected.
# strict - Full SELinux protection.
SELINUXTYPE=targeted
```

References

[1] Asterisk: The Definitive Guide, 3rd Edition
http://www.asteriskdocs.org/

[2] voip-info.org: http://www.voip-info.org

[3] RFC 3261 SIP: Session Initiation Protocol
https://www.ietf.org/rfc/rfc3261.txt

[AsteriskCLI] http://www.asteriskguru.com/tutorials/cli_cmd_14.html.

Glossary

CLI	Command-line interface
CLID	Calling Line Identification
DHCP	Dynamic Host Configuration Protocol
DID	Direct Inward Dialing
DNS	Domain Name Server
HQ	Headquarters
MAC	Media Access Control (address)
PSTN	Public Switched Telephone Network
RCUB	Računarski centar Univerziteta u Beogradu (Belgrade University Computer Centre)
RTP	Real-time Transport Protocol
SIP	Session Initiation Protocol
TCP/UDP	Transmission Control Protocol / User Datagram Protocol (UDP)
TFTP	Trivial File Transfer Protocol
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
VLAN	Virtual Local Area Network
VLAN	Virtual Local Area Network
VoIP	Voice-over-Internet Protocol