Nadace
Vodafone
Česká republika

Voice2Web:Interconnection to mobile network

Petr Vláčil

R&D Centre for Mobile Applications (RDC)
Czech Technical University in Prague

{ vlacipet@fel.cvut.cz }

Table of Contents

1 Purpose of the project

Main scope of the project is to interconnect Vodafone MSC (Mobile Switching Center) situated in RDC laboratory with software PBX (Private Branch eXchange) Asterisk installed on server bolek.feld.cvut.cz. Interconnection is made with E1 link. It consist of 30 voice channels and 1 signaling channel with SS7 signaling system. This connection should allow calls from Vodafone mobile network through Asterisk PBX to its clients connected with VoIP (Voice over Internet Protocol) technology using SIP (Session Initiation Protocol).

Next goal of this project is to configure Asterisk to connect calls from mobile network to VoiceEnabler server at adela.feld.cvut.cz using VoIP. This includes setting a dial-plan and codecs.

2 Used technology

- Mobile switching center Ericsson AXE 10
  - E1 link with SS7 signaling in 16. channel and G.711 A-LAW (PCMA) encoding in voice-channels
- bolek.feld.cvut.cz
  - Software PBX – Asterisk, version 1.4.19.1
  - E1 PCI card – Digium Wildcard TE110P T1/E1 Card, driver version: zaptel 1.4.10
  - Implementation of SS7 signaling for Asterisk – chan_ss7 version 1.0.90
  - Connection to VoiceEnabler adela.feld.cvut.cz – VoIP (SIP + RTP)
- adela.feld.cvut.cz
  - IBM WebSphere VoiceEnabler
- was.feld.cvut.cz
  - IBM WebSphere VoiceServer

3 System architecture

Whole system consists of Vodafone mobile switching center Ericsson AXE 10 connected to server bolek.feld.cvut.cz with E1 link. Bolek is a part of LAN with two additional servers adela.feld.cvut.cz and was.feld.cvut.cz. There is software PBX Asterisk installed on bolek.feld.cvut.cz. IBM WebSphere VoiceEnabler software is installed at adela.feld.cvut.cz. This software controls VoiceXML applications. Next element IBM WebSphere VoiceServer installed on was.feld.cvut.cz is used to recognize voice and play sounds. It communicates with VoiceEnabler. Asterisk connects to VoiceEnabler that starts VoiceXML application. VoiceEnabler redirects RTP traffic that caries sound from Asterisk to the VoiceServer. VoiceServer sends results of voice recognition to VoiceEnabler. Results are used to control VoiceXML application.

4 Network scheme

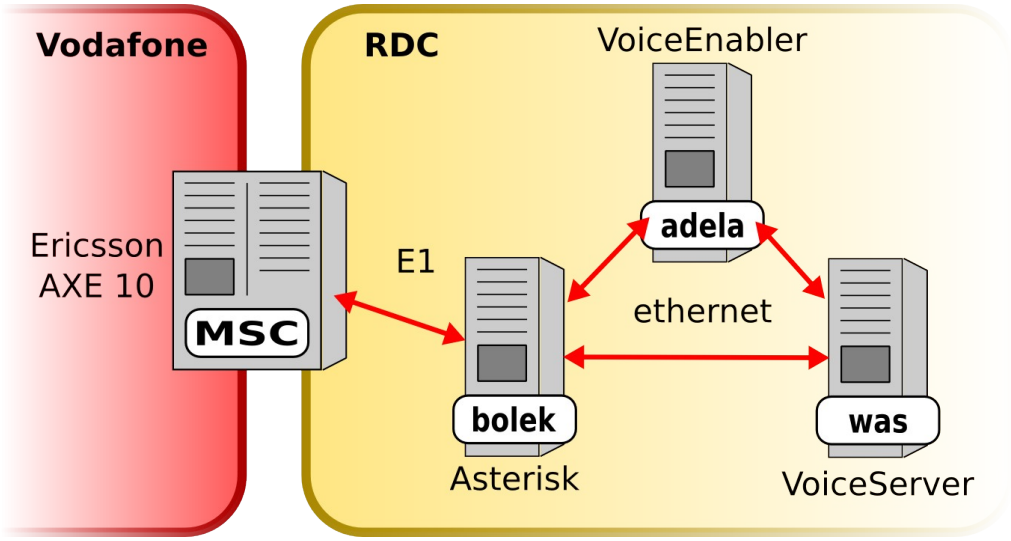

Figure 1: *Network scheme*

5 Solution of the project

At first we need to configure Asterisk PBX installed at bolek.feld.cvut.cz server. Mobile switching center is already configured by Vodafone technicians. Our goal is to configure drivers for Digium Wildcard TE110P T1/E1 PCI card, that is plugged into bolek.feld.cvut.cz. Drivers are called Zaptel drivers and we use version 1.4.19.1. Now we need to configure Asterisk's SS7 signaling channel.

Configuration for card drivers is in a file /etc/zaptel.conf and configuration of SS7 signaling is in a file /etc/asterisk/ss7.conf. Dial-plan can by configured with file /etc/asterisk/extensions.conf.

For communication with VoiceEnabler and VoiceServer Asterisk uses SIP signaling protocol and RTP protocol to transport voice data. Configuration of SIP protocol is in /etc/asterisk/sip.conf

For setting an interconnection we used informations in [1], [2], [3], [4]. Description of configuration parameters is taken from [3], [5] or original configuration files distributed with software packages.

5.1 Configuration of E1 card driver

5.1.1 /etc/zaptel.conf

In detail, configuration of driver is described on many web pages. Description of configuration files was found in original configuration files.

```
span=1,1,0,ccs,hdb3,crc4
```

span=<span num>,<timing source>,<line build out>,<framing>,<coding>

First digit denotes number of the port on the PCI card to which this line relates.

All T1/E1 spans generate a clock signal on their transmit side. The <timing source> parameter determines whether the clock signal from the far end of the T1/E1 is used as the master source of clock timing. If it is,

our own clock will synchronize to it. T1/E1's connected directly or indirectly to a PSTN provider (telco) should generally be the first choice to sync to. The PSTN will never be a slave to you. You must be a slave to it.

Choose 1 to make the equipment at the far end of the E1/T1 link the preferred source of the master clock. Choose 2 to make it the second choice for the master clock, if the first choice port fails (the far end dies, a cable breaks, or whatever). Choose 3 to make a port the third choice, and so on. If you have, say, 2 ports connected to the PSTN, mark those as 1 and 2. The number used for each port should be different.

If you choose 0, the port will never be used as a source of timing. This is appropriate when you know the far end should always be a slave to you. If the port is connected to a channel bank, for example, you should always be its master. Any number of ports can be marked as 0.

Incorrect timing sync may cause clicks/noise in the audio, poor quality or failed faxes, unreliable modem operation, and is a general all round bad thing.

The third digit is the line build-out (or LBO). It is an integer, from the following table. It depends on length of E1 link:

0: 0 db (CSU) / 0-400 m (DSX-1)

1: 400-800 m (DSX-1)

2: 800 m - 1,2 km (DSX-1)

3: 1,2-1,6 km (DSX-1)

4: 1,6-2 km (DSX-1)

5: -7.5db (CSU)

6: -15db (CSU)

7: -22.5db (CSU)

Framing sets the type of frame used on E1 link. CCS (Common Channel Signaling) means that there is one signaling channel for all other voice channels. This signaling channel uses SS7 signaling system. Signalization is carried in 16th channel of E1 link in RDC.

Coding denotes used link code. HDB3 is used

Last entry shows that CRC4 (Cyclic Redundancy Check) is used to verify correct transmission

`bchan=1-31`

Bchan means that all listed channels are treated separately and raw data are available there. No reinterpretation or analysis is done. This means that even if the 16th channel is used for signaling, driver does not inspect it. All data are passed to chan_ss7 which process them. Chan_ss7 is configured in /etc/asterisk/ss7.conf.

```
loadzone = cz
defaultzone=cz
```

This two parameters defines type of tones used in channels. CZ is two-letter code for Czech Republic. All country codes are available in file file zaptel/zonedata.c

5.1.2 /etc/asterisk/ss7.conf

To set right parameters for SS7 signaling you need to edit file /etc/asterisk/ss7.conf. For example you need to specify which channel caries signaling and you also need to set SPC and DPC (Source Point Code and Destination Point Code) which are used for routing of SS7 messages.

Configuration of /etc/asterisk/ss7.conf:

`[linkset-ls1]`

Defines linkset „ls1“. It is a group of links



Nadace Vodafone Česká republika

`enabled => yes`
Enables linkset

`enable_st => no`
The end-of-pulsing (ST) is not used to determine when incoming address is complete. This is used only when R2 signaling is used

`use_connect => yes`
Reply incoming call with CON rather than ACM and ANM.

`hunting_policy => even_mru`
This sets the hunting policy, ie. the algorithm used to allocate a circuit for outgoing calls. This should be configured appropriately at each end of the SS7 link to minimize the risk of call collision (both ends try to make an outgoing call on the same circuit at the same time). Possible values are **odd_lru** (allocate odd CICs in preference to even CICs, and within each group pick the least recently used CIC); and **even_mru** (allocate even CICs in preference to odd CICs, and within each group pick the most recently used CIC). The default is **odd_lru**.

`context => ss7in`
Incoming calls are placed in the ss7in context in the asterisk dial-plan.

`language => en`
Sets the language for this context.

`subservice => auto`
The subservice field: national (8), international (0), auto or decimal/hex value. The auto means that the subservice is obtained from first received SLTM.

`[link-l1]`
Defines link “l1”

`linkset => ls1`
This link belongs to linkset “ls1”.

`channels => 1-15,17-31`
The speech/audio circuit channels on this link. This numbers are always less then or equal to 31 and more then or equal to 1. Channel 0 is used for synchronization.

`schannel => 16`
The signaling channel.

`firstcic => 1`
The first CIC (Circuit Identification Code). It identifies globally whole channels in the linkset. Each channel in linkset has it's own CIC. First CIC in this link is 1. Last CIC is 31. If we have more links in this linkset, other link would start with 32 for synchronization which is not used. First valid CIC in next link is then 33.

`enabled => yes`
Link is enabled.

`[host-bolek.feld.cvut.cz]`
This part configures computer on which the signaling card is installed. Chan_ss7 auto-configures by matching the machines host name with the host-<name> section in the configuration file. This computer is 'bolek.feld.cvut.cz'. This allow to have one configuration file copied on many hosts. Each host will take its own part of configuration file according to host-name.

`enabled => yes`
The host is enabled.

`opc => 0x326`
Originating Point Code. SS7 Signaling Point Code of this computer

`dpc => ls1:0x7d1`

Destination Point Code. SS7 Signaling Point Code of MSC that we want to connect to. We have to set DPC for every link-set.

```
links => l1:1
```

This assigns defined links to ports of the Digium card.

5.2 Description of communication during a call:

Asterisk communicates with Vodafone MSC through E1 link without any problems. Communication was tested by making phone-calls from mobile phone to Echo application in Asterisk. This tests voice channels in both directions. After hearing a tone, you can say what ever you want and Asterisk will record it. Then press #. Asterisk recognizes a DTMF tone and replay you back the recorded speech. This echo application is created in dial-plan:

```
exten => echo,1,Answer
exten => echo,n,Wait(2)
exten => echo,n,Record(asterisk-recording%d:ulaw)
exten => echo,n,Wait(2)
exten => echo,n,Playback(${RECORDED_FILE})
exten => echo,n,System(/bin/rm /var/lib/asterisk/sounds/${RECORDED_FILE}.ulaw)
exten => echo,n,Wait(2)
exten => echo,n,Hangup
```

Calls from Asterisk to VoiceEnabler was tested using SIP clients registered at Asterisk. SIP channel for Asterisk is configured with /etc/asterisk/sip.conf file. There are two codecs allowed in /etc/asterisk/sip.conf, G.711 μ -LAW (PCMU) and G.711 A-LAW (PCMA). PCMU is written first so it has priority before PCMA. Calls were connected properly and voice recognition also worked. Sound generated at VoiceServer was loud and clear. Now we want to connect this two things together. We want to call from mobile phone to Asterisk and we want Asterisk to connect this call using SIP and RTP to VoiceEnabler.

Calls from mobile network to VoiceEnabler are routed through Asterisk. To allow this we need to set proper dial-plan. Dial-plan will be discussed later. After setting a dial-plan we can make a test phone-call from mobile phone to VoiceEnabler. Test call that we made was connected properly but we did not hear any sounds. To solve this problem we need to take closer look at signaling process. We use Wireshark for this kind of job.

5.2.1 Test call from SIP client to VoiceEnabler

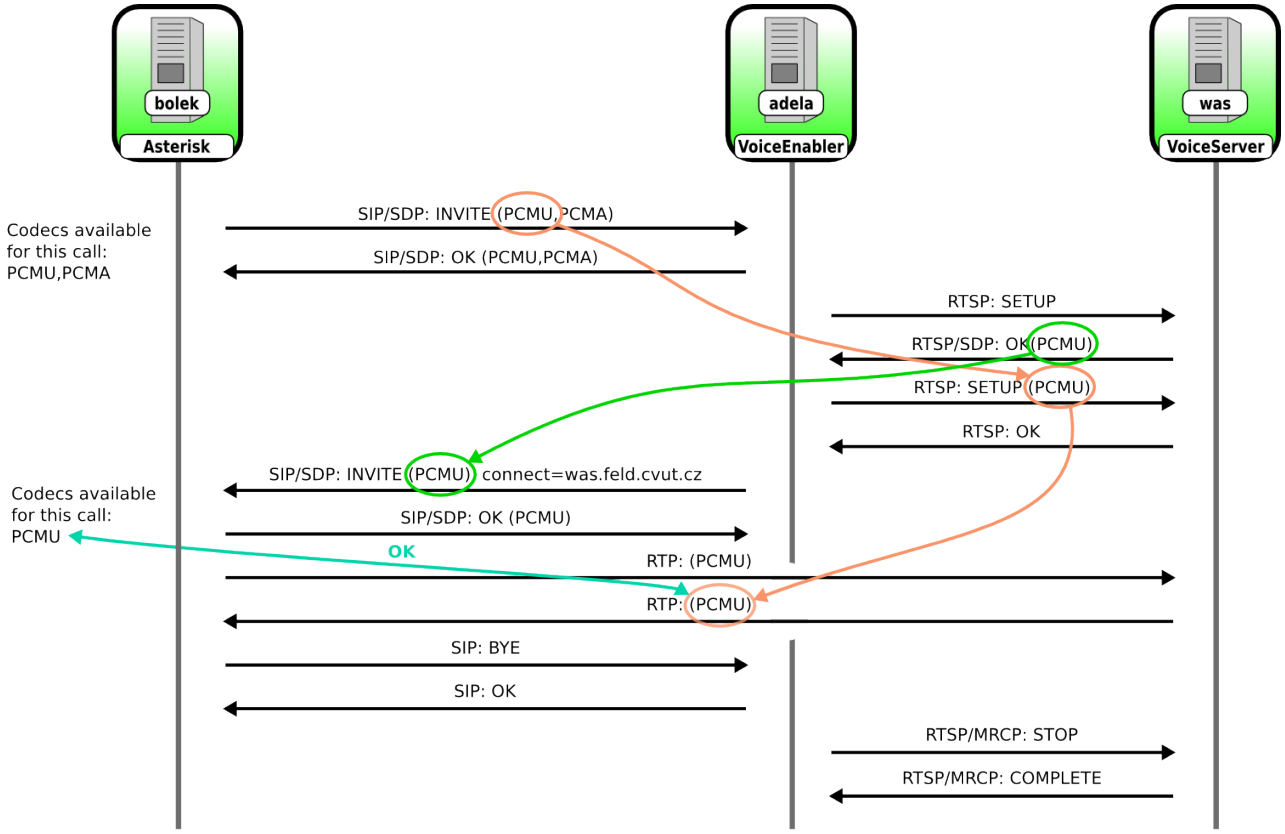

Figure 2: *Call from SIP client*

For testing a call from SIP client we used Linphone. This SIP client was registered at Asterisk. After dialing a number of any service at VoiceEnabler. Asterisk connects a call to VoiceEnabler. Call was successfully made. Scheme of the communication is on the figure. When call is started by SIP client, Asterisk sends SIP message INVITE with SDP (Session Description Protocol) that describes voice session. One of the parameters are codecs that Asterisk supports. VoiceEnabler responds with OK message with its supported codecs. Each side sends codecs that it supports and when it receives RTP stream with voice, it can contain voice in any of these codecs. Now Asterisk can receive audio in PCMU or PCMA. PCMU has priority because it is sent as first in row. VoiceEnabler can receive audio also in PCMU or PCMA.

After session is started between Asterisk and VoiceEnabler. VoiceEnabler contacts the VoiceServer at was.feld.cvut.cz. Communication between VoiceEnabler and VoiceServer is performed by RTSP (Real Time Streaming Protocol). To describe an audio session from Asterisk that is going to be redirected from VoiceEnabler to VoiceServer, VoiceEnabler uses SDP. To control VoiceServer according to some VoiceXML application VoiceEnabler uses RTSP with MRCP (Media Resource Control Protocol). After starting a session between VoiceEnabler and VoiceServer with RTSP:SETUP message VoiceServer sends RTSP with SDP to say that it wants to receive data using PCMU codec. VoiceEnabler restarts the session using another SETUP message with SDP. Codec that is send ind SDP is the first that is stated in first SIP message from Asterisk. In this case it is PCMU. It means that Asterisk want to receive data in PCMU codec. VoiceServer agrees with RTSP:OK message. Now communication between VoiceServer and VoiceEnabler is established and VoiceServer is prepared to receive data from Asterisk in PCMU codec and it will do a voice recognition. It will send a response to Asterisk with PCMU codec.

To complete the connection between Asterisk and VoiceServer VoiceEnabler needs to say to Asterisk that it

should send voice traffic to VoiceServer using codec that VoiceServer expects (PCMU). That is why VoiceEnabler "re-invites" Asterisk. New SIP:INVITE message from VoiceEnabler to Asterisk contains SDP with PCMU codec desired from VoiceServer. It also contains information saying that Asterisk should send voice traffic to VoiceServer. Asterisk responds with SIP:OK message with SDP saying that it now also expects PCMU a starts to send audio to VoiceServer.

Now voice traffic is sent between Asterisk and VoiceServer. SIP signaling is performed between Asterisk and VoiceEnabler and voice recognition data are sent using RTSP with MRCP between VoiceEnabler and VoiceServer.

5.2.2 *Call from GSM mobile network to VoiceEnabler*

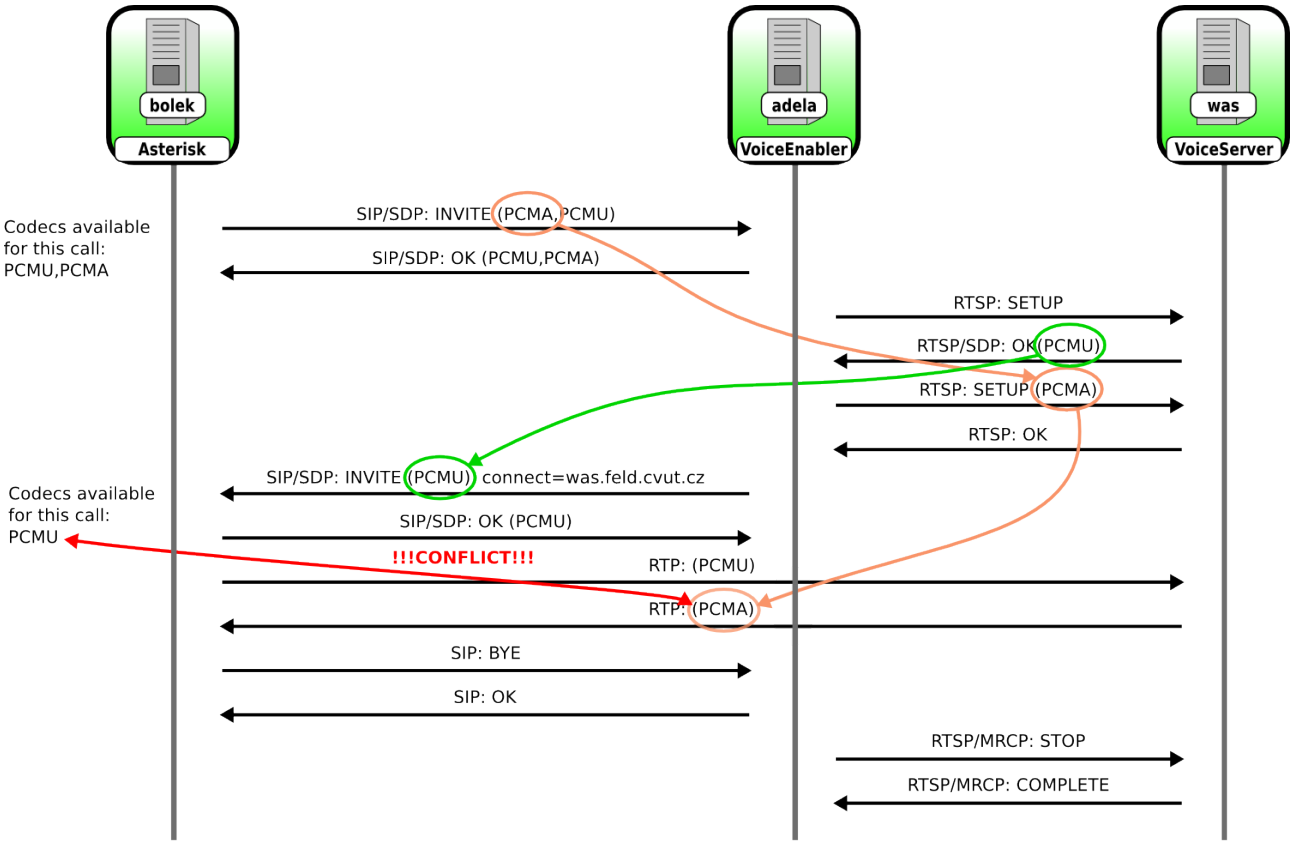

Figure 3: *Call from mobile-phone*

Call from mobile phone is similar to call from SIP client. Only difference is that call arrives to asterisk through E1 link. Following procedures are same as call from SIP client. In closer look we found differences in offered codec. E1 link uses PCMA codec. Asterisk know that in configuration there is allowed to use PCMU and PCMA. Asterisk want to make calls as low demanding on system as possible to save processing power. Primary codec selected in /etc/asterisk/sip.conf is PCMU. However asterisk does not want to convert from PCMA to preferred PCMU because PCMA is also allowed in /etc/asterisk/sip.conf. When sending first SIP:INVITE message Asterisk therefore sends PCMA as preferred codec to try to avoid codec translations and as the second codec offers PCMU. Asterisk is now ready to receive PCMA and PCMU. VoiceEnabler prefers to receive PCMU and then PCMA and sends SIP:OK message with this information.

Session between Asterisk and VoiceEnabler is started. Then VoiceEnabler starts the session with

VoiceServer. VoiceServer offers only PCMU so it expects to receive PCMU. VoiceEnabler takes preferred codec from Asterisk and offers it to VoiceServer. In this case it is PCMA. VoiceServer agrees to send PCMA and session between VoiceServer and VoiceEnabler is started. Now VoiceEnabler has to re-invite Asterisk to redirect voice traffic to VoiceServer. VoiceEnabler therefore sends SIP:INVITE message with redirection to VoiceServer and with codec that VoiceServer expects. This means PCMU. Asterisk agrees with SIP:OK message. Problem is that this message confirms only PCMU codec and Asterisk now expects to receive only PCMU. VoiceServer is informed by VoiceEnabler that Asterisk wants to receive PCMA so it sends traffic in PCMA codec. However Asterisk now supports only PCMU and does not support PCMA for this session. Asterisk does not recognize audio informations and voice channel is quiet.

5.3 Solution of codec negotiation problem

5.3.1 Emergency solution

First and the easiest solution that worked was to ban PCMA codec in SIP configuration. This makes Asterisk to offer only PCMU codec during SIP negotiation. There can not arise problem with PCMA codec. Problem is that you can use only PCMU codec for communication among all clients registered to Asterisk. This is part of the configuration file showing how to allow or disallow coded for whole SIP channel:

```
[general]
disallow=all                    ; First disallow all codecs
allow=ulaw                      ; Allow codecs in order of preference
; allow=alaw                    ; This codec is commented out so it is not used
```

5.3.2 Final solution

This solution uses Asterisk ability to configure codec per user. This settings is done in /etc/asterisk/sip.conf. We have to set PCMU for all calls to host adela.feld.cvut.cz. This allows to have any codec on the system for use among clients and always use PCMU when communicating with VoiceEnabler.

You can see part of configuration file where host adela.feld.cvut.cz is defined and proper codecs are set.

```
[adela.feld.cvut.cz]
type=peer
host=adela.feld.cvut.cz
disallow=all
allow=ulaw
```

6 Codec translation

To list all available codecs you need issue a command `core show codecs`. Our system supports these codecs:

```
         INT    BINARY         HEX    TYPE        NAME    DESC
-------------------------------------------------------------------------------
           1  (1 <<  0)       (0x1)   audio        g723    (G.723.1)
           2  (1 <<  1)       (0x2)   audio         gsm    (GSM)
           4  (1 <<  2)       (0x4)   audio        ulaw    (G.711 u-law)
           8  (1 <<  3)       (0x8)   audio        alaw    (G.711 A-law)
          16  (1 <<  4)      (0x10)   audio    g726aal2    (G.726 AAL2)
          32  (1 <<  5)      (0x20)   audio       adpcm    (ADPCM)
          64  (1 <<  6)      (0x40)   audio        slin    (16 bit Signed Linear PCM)
         128  (1 <<  7)      (0x80)   audio       lpc10    (LPC10)
         256  (1 <<  8)     (0x100)   audio        g729    (G.729A)
         512  (1 <<  9)     (0x200)   audio       speex    (SpeeX)
```

```
      1024 (1 << 10)     (0x400)  audio       ilbc    (iLBC)
      2048 (1 << 11)     (0x800)  audio       g726    (G.726 RFC3551)
      4096 (1 << 12)    (0x1000)  audio       g722    (G722)
     65536 (1 << 16)   (0x10000)  image       jpeg    (JPEG image)
    131072 (1 << 17)   (0x20000)  image        png    (PNG image)
    262144 (1 << 18)   (0x40000)  video       h261    (H.261 Video)
    524288 (1 << 19)   (0x80000)  video       h263    (H.263 Video)
   1048576 (1 << 20)  (0x100000)  video      h263p    (H.263+ Video)
   2097152 (1 << 21)  (0x200000)  video       h264    (H.264 Video)
```

Here you can see list of available codec in our system. Problem is that anyone you call would like use different codecs. Asterisk have to know how to translate one codec to another to be able to interconnect users using different codecs. To display available translations you have to issue command `core show translations`. This is output of our system.

```
      Translation times between formats (in milliseconds) for one second of data
          Source Format (Rows) Destination Format (Columns)

           g723 gsm ulaw alaw g726aal2 adpcm slin lpc10 g729 speex ilbc g726 g722
       g723    -   -    -    -        -     -    -     -    -     -    -    -    -
        gsm    -   -    2    2        2     2    1     3    -     -    -    2    -
       ulaw    -   2    -    1        2     2    1     3    -     -    -    2    -
       alaw    -   2    1    -        2     2    1     3    -     -    -    2    -
    g726aal2   -   2    2    2        -     2    1     3    -     -    -    1    -
      adpcm    -   2    2    2        2     -    1     3    -     -    -    2    -
       slin    -   1    1    1        1     1    -     2    -     -    -    1    -
      lpc10    -   3    3    3        3     3    2     -    -     -    -    3    -
       g729    -   -    -    -        -     -    -     -    -     -    -    -    -
      speex    -   -    -    -        -     -    -     -    -     -    -    -    -
       ilbc    -   -    -    -        -     -    -     -    -     -    -    -    -
       g726    -   2    2    2        1     2    1     3    -     -    -    -    -
       g722    -   -    -    -        -     -    -     -    -     -    -    -    -
```

As you can see our system does not know how to translate for example from speex to any other codecs. It can just pass this codec between users but can not translate it to any other codec. Number on intersections are time in milliseconds that it takes to translate one second of voice in codec on row to codec in column.

7 Setting a dial-plan

Dial-plan for Asterisk is set in extensions.conf. Whole dial-plan is divided into so-called contexts. Each context is group of routing rules that are somehow related to each other. For example all calls that are made by our SIP clients are grouped in context `from_sip`. This contexts also provide some level of security. If we want to provide some of our users with calls to payed numbers, we could put routes to this numbers into e.g. `payed` context and then allow only selected users to use this context.

Dial-plan can be divided into more files. These files can be included into extensions.conf using `#include "file-name"` statement. Files can be included many-times into extensions.conf but only once per context. It does not make sense to put same routing rules to context two-times.

Routing rules are called extensions. Syntax of this extension is described on following example. This is extension from `from_ss7` context. When anyone in the Vodafone network calls 779999222 or +420779999222 Asterisk answers the call, then dial SIP/adela.feld.cvut.cz/777. After the call with SIP/adela.feld.cvut.cz/777 is finished it ends the call. Calls from E1 link need Answer() otherwise call is not successful. Calls from SIP does not need Answer().



Every extension in dial-plan starts with exten => followed by extension number. Next entry is priority. Asterisk uses this number to determine order of lines. This means that extension is not processed line by line but according to this number. Instead of numbering each line you can use “n” which represents number of previous line + 1. This provides easy insertion of lines without renumbering following lines. After priority comes Asterisk's applications. These applications perform operations with call, allow conditional jumps and so on. For description of these applications see [1].

```
exten => 779999222,1,Answer()
exten => 779999222,n,Dial(SIP/adela.feld.cvut.cz/777)
exten => 779999222,n,Hangup()
```

7.1 Numbering

We decided to use 1000 – 1999 numbers in our system. This range was divided into smaller parts as it will be described later. We also have E1 link connected to our Asterisk system. This allow us to make calls from and to Vodafone mobile network. We have range of thirty numbers available on this link.

1001 – 1003	Three testing numbers to test calls to Asterisk (hello-world, info, echo)
1080 – 1099	Numbers of SIP users.
1100	Number of Call Through simple menu allowing to call any application in range 1100 – 1999.
779999201 – 779999230	Testing numbers available from Vodafone network.

For our purposes we use these contexts:

default	This is default context used in Asterisk dial-plan. It should not be used for normal call routing.
from_sip	Routes calls among SIP users and allows them to call testing applications on VoiceEnabler.
from_ss7	Routes calls from Vodafone network to some SIP accounts and to application selection menu, that allow calling VXML applications at VoiceEnabler.
testing	This context provides extensions that can be used to test calls to Asterisk. You can test your microphone by using echo application. Testing context was created to ease adding testing functionality to any other context. You can use testing extension from any other context by adding e.g. `Goto(testing,echo,1)` to your dial-plan route.
dynamic_apps_testing	Provides mentioned simple menu for selection of VXML applications on VoiceEnabler. This menu is constructed with DTMF detector. Calling person enters extension number by pressing keys on phone keyboard. Asterisk is detecting tones and tries to match entered numbers to available extensions. As soon as enough digits are entered, extension is dialed. This introduces security risk. User can enter any extension available. If we have available any extension to call out of our system, user can call out through our simple menu. That is why we created separated context for allowing only to choosing desired applications. In this context we can call only VoiceEnabler. Call is redirected to this context whenever user from SIP or from E1 link dials number of menu application.

7.2 Call Through

Call Through provides ability to directly call users that are behind some PBX. In our system we set simple menu that is represented by one phone number. After calling this number, Asterisk will ask you to enter a number of extension to call. DTMF detector detects pressed numbers and Asterisk tries to find extension to call.

This simple menu consist of starting extension “s” that plays menu sound and waits for dialed numbers.

```
exten => s,1,Wait(1)
exten => s,n,Background(vm-enter-num-to-call)
exten => s,n(dtmf_detector),WaitExten()
```

After detecting a digits asterisk tries to match them to any extensions. This extension is represented by pattern matching extension _1ZXX that makes Asterisk to call to applications at VoiceEnabler from range 1100 to 1999. After call to VoiceEnabler is finished, user gets congestion tone for 2 seconds.

```
exten => _1ZXX,1,Dial(SIP/adela.feld.cvut.cz/${EXTEN},,g)
exten => _1ZXX,n,Playtones(congestion)
exten => _1ZXX,n,Wait(2)
exten => _1ZXX,n,Hangup()
```

Extesion “t” and “i” take care of timeout and invalid dialed extensions. When wrong extension is dialed everything starts from beginning. System warns user that dialed extension was incorrect an waits for another try. Default timeout of WaitExten() application is 10 seconds. After passing the timeout system will say “goodbye” and hangs up the call.

```
exten => i,1,Background(pbx-invalid)
exten => i,n,WaitExten()

exten => t,1,Playback(vm-goodbye)
exten => t,n,Hangup()
```

8 Conclusion

In first part of our project we have to configure interconnection of Vodafone MSC Ericsson AXE 10 and software PBX Asterisk using E1 link and PCI card Digium Wildcard TE110P T1/E1. Many materials on the Internet can be found on this theme. Interconnection was successful without much effort.

Other part of the project was setting Asterisk so it was able to interconnect calls from mobile-network to VoiceEnabler and VoiceServer. Problem with codecs negotiation was found. Solution of this problem involved a lot of packet tracing an analyzing. Two solutions was found. First solution was allowing only PCMU codec on Asterisk. This solution was only a first idea an is not very usable because other codecs would be banned for whole system. Second solution was to set SIP channel at Asterisk so that it offers only PCMU to VoiceEnabler. Other users of the system can use all other available codecs.

Third part of the project was fine-tune Asterisk's dial-plan. In this part we renumbered SIP clients into range of 1080-1099. New architecture was introduced to dial-plan and DTMF Call Through menu was implemented. This menu allows to call all applications available at VoiceEnabler from Vodafone network.

9 References

1. Jim Van Meggelen, Leif Madsen, Jared Smith: Asterisk, The Future of Telephony, 2nd edition, O'Reilly, 2007, ISBN 0596510489
2. Petr Kubeš: Signalizace SS7 v ústředně Asterisk, Master Thesis, ČVUT, 2008
3. VOIP Wiki - a reference guide to all things VOIP, 2008,URL <http://www.voip-info.org>
4. Jan Rudinský, Miroslav Vozňák, Jan Růžička: Asterisk and SS7, CESNET, 2006, URL <http://www.cesnet.cz/doc/techzpravy/2006/asterisk-ss7/>
5. Chan_ss7 documentation: ss7.conf.template.single-link, Sifira A/S, 2008

10 Annex A

10.1 The Phone book

10.1.1 Calls from mobile network

Extension	Asterisk Dial() string: type/identifier[/extension]
779 999 201	SIP/adela.feld.cvut.cz/1101
779 999 202	SIP/adela.feld.cvut.cz/1102
779 999 203	SIP/adela.feld.cvut.cz/1103
779 999 204	SIP/adela.feld.cvut.cz/1104
779 999 205	SIP/adela.feld.cvut.cz/1105
779 999 211	SIP/honza
779 999 212	SIP/petr
779 999 222	Call Through menu for applications (range 1100-1999)
779 999 229	Hello-world testing
779 999 230	Echo testing

10.1.2 Calls from SIP clients to numbers

Extension	Asterisk Dial() string: type/identifier[/extension]
1001	Hello-world testing
1002	Info about Asterisk testing
1003	Echo testing

Extension	Asterisk Dial() string: type/identifier[/extension]
1080	SIP/honza
1081	SIP/pavel
1082	SIP/tomas
1083	SIP/lukas
1084	SIP/xavi
1085	SIP/petr
1086	SIP/laurent
1087	SIP/roman
1088	
1089	
1090	SIP/test
1091	SIP/test1
1092	
1093	SIP/test3
1094	
1095	SIP/test5
1096	
1097	SIP/test7
1098	
1099	SIP/test9
1100	Call Through menu for applications (range 1100-1999)
1101-1999	Applications at VoiceEnabler

10.1.3 Calls from SIP clients to names

Extension	Asterisk Dial() string: type/identifier[/extension]
hello-world	Hello-world testing
info	Info about Asterisk testing
echo	Echo testing
honza	SIP/honza
pavel	SIP/pavel
tomas	SIP/tomas
lukas	SIP/lukas
xavi	SIP/xavi
petr	SIP/petr
laurent	SIP/laurent

Extension	Asterisk Dial() string: type/identifier[/extension]
roman	SIP/roman
test	SIP/test
test1	SIP/test1
test3	SIP/test3
test5	SIP/test5
test7	SIP/test7
test9	SIP/test9

11 Revisions

Revision	Author	Date	Notes
1.0	Petr Vláčil	25.8.2008	First release
1.1	Petr Vláčil	1.9.2009	Change name of the project from ACC to Voice2Web. Added Revisions section.