VIDEO GATEWAY

Installation and Configuration Instructions

1 Introduction

This document is the response to the request for quotation for the implementation and deployment of a video gateway with asterisk.

1.1 References

[1] Medooze
http://www.medooze.com

[2] Asterisk
http://www.asterisk.org

2 Technical solution

2.1 Architecture

The following diagram shows the architecture of the final solution that will enable a 3G mobile handset to establish a UMTS video call with a Video Softphone IP.

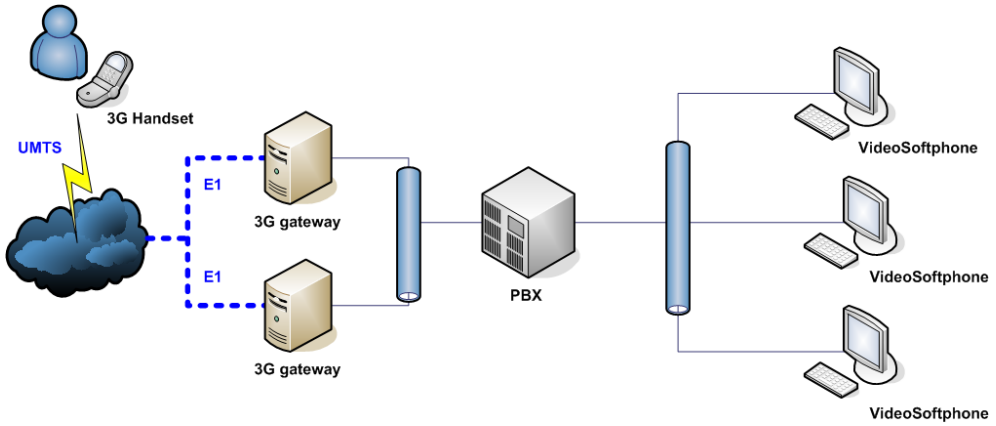

Figure 1 Technical Architecture

The 3G Gateway will be connected to a ISDN E1 interface and with an IP PBX. It will handle both the SIP and H245 negotiations to establish the two legs of the communication, one with the 3G handset and the other with the Video Softphone. When the communication is establish it will mux and demux the h223 video call data from the 3G handset to be able to bridge the audio and video between the endpoints performing the necessary transcoding operations needed to meet the endpoint capabilities.

2.2 Technical design

The 3G gateway will be implemented as an application to the Asterisk PBX.

The app_h324m[1] application will allow to bridge the h223 multiplexed channel to a local asterisk channel that will be routed again to the pbx core following the configured dialplan.

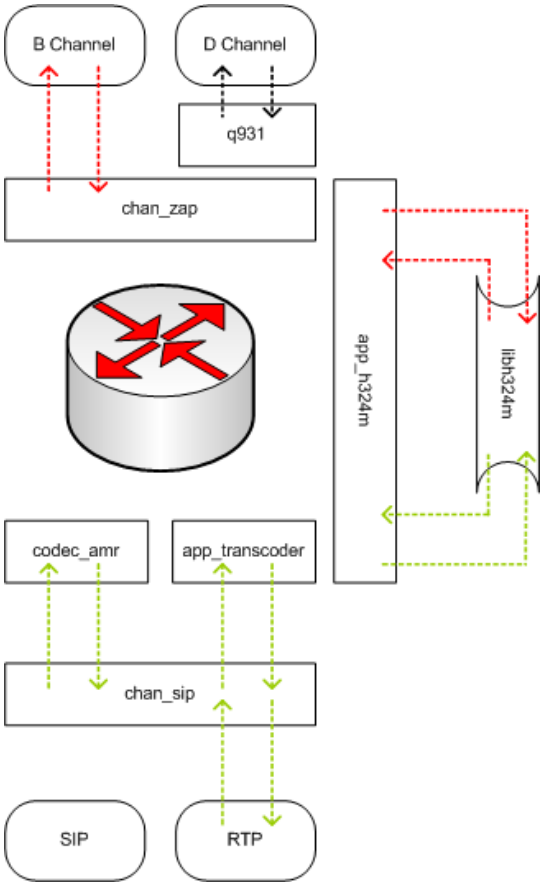

Figure 2 Technical design

The library will handle the h245 negotiation with the 3G handset and establish both audio and video streams.

Media streams will be bridged together and forwarded from one channel to another, multiplexing or demultiplexing in each case.

Video calls require a special media formats and codecs. Audio will be passed through an AMR codec to encode it before sending to the 3G handset and to decode the reverse stream to allow asterisk to handle it. Video sent to the 3G handset needs to be rescaled to QCIF size and transcoded to fit in 52Kbits.

3 Prequisites

The following packages must be installed before the installation procedure is run.

```
#
#Installing prerequisites
#

yum install patch
```

Also no precompiled packages from pwlib or ffmpeg must be present on the system before the installation procedure is run.

All software source code will be installed in `/usr/local/src` if the directory is not present it should be created.

The library path for library installation will be /usr/local/lib so ti should be added to the system library path if not already done in the `/etc/ld.so.conf` config file.

Asterisk 1.4 source code, original or modified version must be installed on `/usr/local/src/asterisk` directory.

4 Installation procedure

4.1 Source downloading

The first step is to download the source code for the components needed.

The external software used is:

- mpeg4ip version 1.5.0.1
- amrnb lib version 7.0.0
- ffmpeg latest revision
- libtool library
- bison
- patch
- SDL development library

The following script downloads and decompresses the source code in the directories for installing.

Script:

```
#
# Downloading source code
#

mkdir -p /usr/local/src
cd /usr/local/src

wget http://downloads.sourceforge.net/mpeg4ip/mpeg4ip-1.5.0.1.tar.gz
wget http://ftp.penguin.cz/pub/users/utx/amr/amrnb-7.0.0.0.tar.bz2
wget http://downloads.sourceforge.net/openh323/ptlib-v1_12_0-src.tar.gz

tar xvzf mpeg4ip-1.5.0.1.tar.gz
tar xvzf ptlib-v1_12_0-src.tar.gz
bunzip2 amrnb-7.0.0.0.tar.bz2
tar xvf amrnb-7.0.0.0.tar

mkdir zips
mv amrnb-7.0.0.0.tar mpeg4ip-1.5.0.1.tar.gz ptlib-v1 12 0-src.tar.gz zips

#
# Subversion repository checkout
#

svn checkout svn://svn.ffmpeg.org/ffmpeg/trunk ffmpeg
svn co https://asteriskvideo.svn.sourceforge.net/svnroot/asteriskvideo asteriskvideo
git clone git://git.videolan.org/x264.git
```

4.2 Library Installation

4.2.1 MPEG4IP

To configure, compile and install the mpeg4ip libraries run the following script in the source directory.

Script:

```
#
# Compiling Mpeg4IP
#

cd mpeg4ip-1.5.0.1/
./bootstrap && ./configure --disable-player --disable-server
make && make install
cd ..
```

4.2.2 AMR NB

To configure, compile and install the amr nb libraries run the following script in the source directory.

Script:

```
#
# Compiling AMR NB 7.0
#

cd amrnb-7.0.0.0
./configure
make && make install
cd ..
```

4.2.3 X264

To configure, compile and install the x264 encoder libraries run the following script in the source directory.

Script:

```
#
# Compiling FFMPEG
#

cd x264
./configure && make && make install
cd ..
```

4.2.4 FFMPEG

First apply the following patch in the libavcodec directory that will change the

```
Index: libx264.c
===================================================================
--- libx264.c   (revision 21036)
+++ libx264.c   (working copy)
@@ -160,6 +160,12 @@
     x4->params.rc.i_vbv_buffer_size = avctx->rc_buffer_size / 1000;
     x4->params.rc.i_vbv_max_bitrate = avctx->rc_max_rate    / 1000;
     x4->params.rc.b_stat_write      = avctx->flags & CODEC_FLAG_PASS1;
+
+    /********* SERGIO ****/
+    x4->params.i_slice_max_size            = 1350;
+    x4->params.rc.i_lookahead       = 0;
+    /*********************/
+
     if (avctx->flags & CODEC_FLAG_PASS2) {
         x4->params.rc.b_stat_read = 1;
     } else {
```

To configure, compile and install ffmpeg and libavcodec libraries run the following script in the source directory.

Script:

```
#
# Compiling FFMPEG
#

cd ffmpeg
./configure --enable-shared --enable-gpl --enable-nonfree --enable-libgsm --
enable-libx264 && make && make install
cd ..
```

4.2.5 PTLIB

Script:

```
#
# Compiling PTLib V1.12.0
#

cd ptlib v1 12 0/
./configure
make && make install
cd ..
```

4.2.6 LIB H324M

Script:

```
#
# Compiling LIB H324M
#
```

```
cd asteriskvideo/libh324m/
make && make install
cd ..
```

4.3 ASTERISK

4.3.1 Prerequisites

Script:

```
#
# Prepare Asterisk
#

cd asterisk
./configure
```

4.3.2 Codec AMR

Script:

```
#
# Compiling Codec AMR
#

cd codecs
patch -p0 < /usr/local/src/asteriskvideo/amr/amr-asterisk-patch.txt
cd codecs
ln -s /usr/local/src/asteriskvideo/amr/amr_slin_ex.h .
ln -s /usr/local/src/asteriskvideo/amr/slin_amr_ex.h .
ln -s /usr/local/src/asteriskvideo/amr/codec_amr.c .
mkdir amr
cd amr
wget http://www.3gpp.org/ftp/Specs/archive/26 series/26.104/26104-700.zip
unzip -j 26104-700.zip
unzip -j 26104-700_ANSI_C_source_code.zip
ln -s /usr/local/src/asteriskvideo/amr/Makefile .
cd ../../
```

4.3.3 Applications

Script:

```
#
# Compiling Codec AMR
#

cd apps
ln -s /usr/local/src/asteriskvideo/app_mp4/app_mp4.c .
ln -s /usr/local/src/asteriskvideo/app_transcoder/app_transcoder.c .
ln -s /usr/local/src/asteriskvideo/app_rtsp/app_rtsp.c .
ln -s /usr/local/src/asteriskvideo/app h324m/app h324m.c .

echo "app mp4.so : app mp4.o" >> Makefile
echo -e "\t\$(CC) \$(SOLINK) -o \$@ \${CYGSOLINK} \$< \${CYGSOLIB} -lmp4 -
lmp4v2 " >> Makefile

echo "app h324m.so : app h324m.o" >> Makefile
echo -e "\t\$(CC) \$(SOLINK) -o \$@ \${CYGSOLINK} \$< \${CYGSOLIB} -lh324m
```

```
" >> Makefile

echo "app_transcoder.so : app_transcoder.o" >> Makefile
echo -e "\t\$(CC) \$(SOLINK) -o \$@ \${CYGSOLINK} \$< \${CYGSOLIB} -lavcodec -
lswscale" >> Makefile

cd ..
```

4.3.4 Build and install

Script:

```
#
# Compile and Install
#

make && make install
```

4.4 Configuration

4.4.1 Codec AMR

Append the following lines to `codecs.conf` in order to configure the amr codec properly:

```
[amr]
octet-aligned=1
```

4.4.2 Dialplan

To configure the dialplan to accept an incoming 3G call, in the dialplan context configured for the ISDN channel configure the following:

```
[default]
exten => 5002,1,H324m_gw(test@3g)

[3g]
exten => test,1,H324m_gw_answer()
exten => test,n,Video_loopback()
```

Transcoding

To configure the dialplan to accept an incoming call, in the dialplan context configured for the SIP incoming calls configure the following:

```
[default]
include => transcode

[transcode]
exten => 1,1,Answer()
exten                                                                    =>
1,n,transcode(,1@camera,h263@cif/fps=30/kb=800/qmin=1/qmax=12/gs=1000)


[camera]
exten => 1,1,Answer()
exten => 1,n,rtsp(rtsp://192.168.0.217/live.sdp)
```

The first `transcode` section will answer the call and create a pseudo channel that will start in the extension `1` of the section `camera`.

The transcode application wills transcoded audio and video from one channel to another based on the parameters specified in the call.

The `camera` section will connect to the rtsp server and grab the audio and video camera and copy it to the asterisk channel.

For further documentation about applications in `app_transcode` and `app_rtsp` modules refer to chapter 4.

IVVR

Audio and video playback and recording can be handled by the applications available in the `app_mp4` module to create interactive voice and video applications:

```
Exten => 1,1,mp4play(/var/3gmenu/incoming/msg.3gp)
exten => 1,n,Goto(start,6);

exten => 2,1,mp4save(/var/3gmenu/incoming/msg.3gp,V#)
exten => 2,n,Hangup()
```

For documentation about applications in `app_mp4` module refer to chapter 4.2.

5 Module documentation

5.1 app_rtsp

5.1.1 rtsp

Establish a rtsp connection.

```
rtsp(url)
```

Description

This function establish a rtsp connection with the rtsp server indicated in the parameter

Examples:

```
[rtsp]
exten => 1,1,Answer()
exten => 1,1,rtsp(rtsp://192.168.1.2/live.sdp)
```

5.2 app_mp4

5.2.1 mp4_save

Records call into an mp4 file.

```
mp4save(filename,[options])
```

Description

The application saves the call audio and video and hit information into the mp4/3gpp file specified.

Note: If you are working with amr it's recommended that you use 3gp as your file extension if you want to play it with a video player.

Available options:
'v': activate loopback of video
'V': wait for first video I frame to start recording
'0'..'9','#','*': sets dtmf input to stop recording

Note: waiting for video I frame also activate video loopback mode.

Examples:

```
; record video to selected file
exten => ._,1,mp4save(/tmp/save.3gp)

; record video and stop on '#' dtmf
exten => . ,1,mp4save(/tmp/save.3gp,#)
```

```
; activate loopback of video
exten => ._,1,mp4save(/tmp/save.3gp,v)

; wait for first videoto start recording
exten => ._,1,mp4save(/tmp/save.3gp,V)

; wait for first videoto start recording
; and stop on '9' dtmf
exten => ._,1,mp4save(/tmp/save.3gp,V9)
```

5.2.2 mp4_play

MP4/3GPP file playblack

`mp4play(filename,[options])`

Description

Plays the audio and video from an 3GPP/MP4 file.

Note: The file MUST be hinted.

Available options:
'n(x)': number of digits (x) to wait for
'S(x)': set variable of name x (with DTMFs) rather than go to extension
's(x)': set digits, which should stop playback

Examples:

```
; play video file to user
exten => ._,1,mp4play(/tmp/video.mp4)

; play video file to user and wait for 3 digits
exten => ._,1,mp4play(/tmp/test.mp4,'n(3)')

; play video file to user and wait for 3 digits and
; set them as value of variable DTMF_INPUT
exten => ._,mp4play(/tmp/test.mp4,'n(3)S(DTMF_INPUT)')

; play video file, wait for 3digits or break on '#'
exten => ._,1,mp4play(/tmp/test.mp4,'n(3)s(#)')
```

5.3 app_transcode

5.3.1 transcode

Video transcode

`transcode(informat|extension@context|outformat)`

Description

This application creates a pseudo channel and places it in the specified context. When both channels are established it performs video transcoding from one channel to another.

The transcoding applied in each direction are specified in the `informat` and `outformat` parameters. If no format is specified, transcoding is performed on that direction, just media copy. Audio is copied unmodified.

Only h263p codec is supported for transcoding [rfc 2429] for output, input codecs are h263/h263 and mpeg4.

Transcoding format:
`codec@size/[param/param/...]`

Available codecs:
h263

Available sizes:
QCIF or CIF

Available params:
`fps`: number of frames per second
`kb`: video bandwidth
`qmin`: Maximum quality [1..31] (lower best)
`qmax`: Minimum quality [1..31] (lower best)
`gs`: Gop size, frame distance between IFrames.

Examples:

```
; Transcode an incoming channel for a 3g compatible call
exten => ._,1, transcode(h263@qcif/fps=7/kb=52/qmin=12/qmax=31/gs=50,${EXTEN}@3g-call,)
```