How-to: home video monitoring

dernière modification par Elisa Nectoux

le 2020/11/06 18:07

Contents

Introduction

The linphone software fits well in embedded systems, which makes it a good candidate for being used in home automation devices where video is to be capture or displayed. However a SIP user-agent ifself is not sufficient for setting up a fully functional SIP network: we propose the use of Flexisip, a SIP proxy server also able to run with reduced footprint on embedded devices as well as large scale cloud deployment.

In this document we will describe how to setup a home automation SIP network, with user agents located in:

- At the door of the house, an entry panel device equipped with a camera, in which users press a "ring" button

- In the house, home screen devices equipped with a display screen, in which users receive the video and audio of the person at the door who pressed the ring button.

- smartphones, belonging to the owner of the house, with which he can receive calls made from the entry panel, regardless of whether the smartphone is at home or outside, typically connected to a mobile data network (3G/4G). Users can also place monitoring calls to the entry panel from their smartphones.

Two SIP servers are involved in this design:

- a flexisip proxy server at home, running in the entry panel device, further referred in this document as the "**local flexisip**". In the scenario described here, we've decided to run it on the entry panel device, but it could be running in other pieces of hardware provided that it is in the home network. This proxy server has two main roles:

  - routing calls in the house (in a real scenario we could have several home screen in different rooms, all ringing and showing the video at the same time)
  - establishing a secure and both-way authenticated communication with a second flexisip proxy server instance running on public internet.

- the **cloud flexisip** proxy server, running with public IP addresses on a rather big machine (depending on the number of houses and users to be connected). This proxy server is responsible for making the bridge between smartphones located in the public internet, and their associated home network.

Getting the software

For this setup, we've used raspberry-pi 2 devices in order to emulate the in-house devices.

For the user-agents, we'll use the linphone-daemon console tool, which is part of the Linphone software. Linphone-daemon is a console frontend to the liblinphone, reading simple commands and writing command output from and to stdin/stdout or a UNIX pipe. For more information about supported commands, type 'help' in the linphone-daemon's shell prompt.
Of course, developing a custom application based on liblinphone (written in C, C++ or python) that could directly interact with the hardware's buttons / or the screen would be more powerful for a real-world solution.

Compilation

A procedure for compiling linphone on raspberry-pi is available here.
The compilation of flexisip requires the same build environment and can be done as follows:

```
sudo apt-get install libssl-dev
git clone git://git.linphone.org/flexisip.git --recursive
cd flexisip
./prepare.py -DENABLE_PUSHNOTIFICATION=OFF -DENABLE_REDIS=OFF -DENABLE_PROTOBUF=OFF
make -j4
```

Yocto recipes

Alternatively, yocto recipes are available to build an image with flexisip and linphone for ARMv7 devices.

Cloud flexisip

For the cloud server, we recommend a redhat 7 / centos 7 distribution, so that it is easy to install flexisip from our yum repositories .

Configuration

Overview

First, let's give arbitrary hostnames to our devices, to avoid dealing with IP addresses further in this document. We assume that the hostnames are defined for every device, either through /etc/hosts or through a local DNS server of the local router.

- "entry-panel" : the hostname of the entry-panel device
- "home-screen1" : the hostname of a first home screen
- "home-screen2" : the hostname of a second home screen
- "mycloudserver.example.org": the hostname of the "cloud flexisip" server. Obviously this name should be defined in public DNS.

In order to go straitforward to working call flows between devices and smartphones, we'll first bypass most security aspects. Enabling strong authentication and data protection is discussed in next chapter.

Setting up the local network

Local flexisip (in the entry panel)

This proxy server performs routing for calls internal to the house. We need to instruct this flexisip to act as Registrar and Proxy for a local domain that we'll call "myhouse".
A documented configuration file with default values can be generated with "flexisip --dump-default all > flexisip.conf" . The *flexisip.conf* is to be placed within */etc/flexisip/*.
You may edit the default configuration file and change the configuration properties as below, or simply copy/paste them into an empty file.

```
[module::Authentication]
enabled=true
auth-domains=myhouse
db-implementation=file
datasource=/etc/flexisip/passwd
[module::Registrar]
reg-domains=myhouse
static-records-file=/etc/flexisip/routes.conf
[module::MediaRelay]
prevent-loops=false
nortpproxy=disable
```

Additionally we setup a special SIP address *"sip:everyone@myhouse"*, for the purpose of being called by the entry-panel user-agent in order to ring to all screens and smartphone users.
This is to be done in the file */etc/flexisip/routes.conf*, as follows:

```
<sip:everyone@myhouse>   <sip:home-screen@myhouse>, <sip:bob@myhouse>, <sip:alice@myhouse>
```

Here, "bob" and "alice" are the smartphone users.

We need to setup passwords for digest authentication of smartphone users by creating a */etc/flexisip/passwd* file:

```
version:1
bob@myhouse clrtxt:bobsecret ;
alice@myhouse clrtxt:alicesecret ;
home-screen@myhouse clrtxt:abcdefgh ;
entry-panel@myhouse clrtxt:12345678 ;
```

Now you can start flexisip with:

```
flexisip -c /etc/flexisip/flexisip.conf --debug
```

Entry panel user-agent

We have to instruct linphone-daemon to register to the local flexisip as user "*entry-panel*", which can be done by creating the following linphone configuration file:

entry_panel_linphonerc:

```
[sip]
sip_tcp_port=5070
sip_port=5070
default_proxy=0
[sound]
#we turn off software echo cancellation to save CPU processing, assuming it is done by the hardware
echocancellation=0
[proxy_0]
reg_proxy=<sip:127.0.0.1;transport=tcp>
reg_route=sip:127.0.0.1;transport=tcp
reg_identity=sip:entry-panel@myhouse
reg_expires=600
reg_sendregister=1
```

```
[misc]
#this option is to tell liblinphone to send real audio/video when requested early media (by default it sends nothing for security reasons).
real_early_media=1
[auth_info_0]
username=entry-panel
passwd=12345678
realm=myhouse
domain=myhouse
[video]
automatically_initiate=1
automatically_accept=1
```

Linphone-daemon is launched as follows:

```
linphone-daemon --auto-answer -C --factory-config entry_panel_linphonerc --log /tmp/log.txt
```

-C is to specify video Capture mode only.

Home screen user-agent

In a way similar to the entry panel, use linphone-daemon with the following configuration file, to instruct it to register to the local flexisip as "*sip:home-screen@myhouse*".
If several home screens are used, the should have all the same configuration.

home_screen_linphonerc:

```
[sip]
sip_tcp_port=5070
sip_port=5070
default_proxy=0
#this option is to tell liblinphone to propose early media immediately after receiving the call,
#so that the audio and video can be received.
incoming_calls_early_media=1
[sound]
#we turn off software echo cancellation to save CPU processing, assuming it is done by the hardware
echocancellation=0
[proxy_0]
reg_proxy=<sip:entry-panel;transport=tcp>
reg_route=sip:entry-panel;transport=tcp
reg_identity=sip:home-screen@myhouse
reg_expires=600
reg_sendregister=1
[auth_info_0]
username=home-screen
passwd=abcdefgh
realm=myhouse
domain=myhouse
[video]
automatically_initiate=1
automatically_accept=1
displaytype=MSGLXVideo
```

Linphone-daemon should be start as follows:

```
linphone-daemon -D --factory-config home_screen_linphonerc --log /tmp/log.txt
```

-D is to specify video Display mode only.

Smartphone users

We invite to use the linphone application for iOS or Android, that anyone can download for free from Appstore and Google Play.

Run the assistant, select "Use SIP account", and enter the following information:
Username: bob
password: bobsecret
domain: myhouse
transport: tcp

Then validate. The registration will fail because linphone won't know which proxy to connect to for domain "myhouse".
Go to the settings, by clicking on the newly created account, and set the following:

- *proxy server* field should be set to *sip:entry-panel;transport=tcp*
- toggle ON the "*Outbound proxy*" option.
- In video settings, toggle ON "automatically accept" and "automatically initiate" video stream, so that the call start with video from the begining.

Testing the local network

Now you can make calls to validate the home network is fully operational.

Call from the entry panel

Within the linphone-daemon shell prompt, type:

call everyone

Note that the actual sip address being called will be sip:everyone@myhouse. Indeed, liblinphone automatically creates the SIP URI to call by using the domain part of the active SIP account, when provided with a username only, ie not begining with "sip:".
The smartphones and the homes screens should all ring. Additionally, the home screens should display the video of the entry panel's camera.
If the call gets accepted somewhere, all other devices stop ringing and the call is established with the party that accepted it.
On the home screens, the call can be accepted by typing "*answer*" in the linphone-daemon shell prompt.

Notes

- on iOS calls can't be received when the app is running in the background. This is because the local sip server isn't configured to send push notifications to the linphone application. Sending a push notification to an app requires an SSL certificate that is delivered by Apple to the app's developer only.
- the smartphones won't display the video in early media, because the Linphone application doesn't have this feature (but the library has). A real home automation app would have to call linphone_core_accept_early_media() in order to start receiving video while the call is not yet accepted by the user.

Do a surveillance call from the smartphone

Just type "*entry-panel*" in the linphone's phone number input field (a keyboard pops up when you press in it, so that you can type any sip address), and press the call button. The entry-panel user-agent being started with --auto-answer option, the call will be automatically taken by the entry panel and the video will appear on the smartphone.

Bridging the home network with the cloud

In the previous chapter, we've setup the home network so that calls can take place in the house between the different user-agent. In this chapter, we'll now setup cloud flexisip and and necessary options to the local network to allow call flows from and to smartphone being located outside of the home network.

Cloud server configuration

In order for smartphones to receive calls even when they are not at home, we need to setup the flexisip cloud server. The installation of official flexisip packages is documented here.

The configuration provided below sets flexisip as registrar for the wildcard domain, and gives the capability of "SIP domain registrar", ie it can accept registration for an entire domain
(not just single users). We are going to use this facility for the local flexisip to register the entire "myhouse" domain to the cloud flexisip.

Note: the concept of the registration of an entire domain is an usage created by Belledonne Communications, however it is not something documented in any published RFC, though nothing prohibits it.

As a result, the cloud flexisip will be able to receive thousands of domain registrations from different houses, provided of course that each house has a unique SIP domain name.
Note the enablement of the media relay module, which is necessary for the media streams to be forked between the multiple home screens and the smartphones during early media.

```
[global]
transports=sip:mycloudserver.example.org
[module::Registrar]
reg-domains=*
[module::MediaRelay]
enabled=true
nortpproxy=disable
[inter-domain-connections]
accept-domain-registrations=true
```

Local flexisip configuration

We need to instruct the local flexisip to send the domain registration to the cloud flexisp.
This is done setting up a new configuration file, */etc/flexisip/domain-registrations.conf*, with a single line:

```
myhouse      sip:mycloudserver.example.org;transport=tcp
```

This line tells that the local domain 'myhouse' needs to be registered to the proxy sip:mycloudserver.example.org. This will make the cloud flexisip to route incoming call into the local flexisip, and hence entry panel or screens to receive calls from smartphones.
Note that the local flexisip will keep the connection to the remote one up all the time. **Should a network disconnection happen, the registration will be re-establish automatically. The remote flexisip will use this connection (and won't attempt to create a new one) in order to route requests to the local network. This dynamic registration is robust to NAT, and doesn't require any special setting on the home router (no dyndns, no port forwarding)**.

We also need to tell the local flexisip that bob and alice might be outside, by adding a static route in */etc/flexisip/routes.conf* :

```
<sip:bob@myhouse>    <sip:mycloudserver.example.org;transport=tcp>
<sip:alice@myhouse>    <sip:mycloudserver.example.org;transport=tcp>
```

These lines will cause the local flexisip to send INVITEs to the cloud flexisip in order to query for potential devices from alice or bob, when they are called by the entry panel or a screen. These are routes that come in addition to the contact addresses of direct registrations eventually done by bob and alice user-agents.

In order for calls to be efficiently passed from local flexisip to cloud flexisip, it is recommended to enable *target factorization*.
This feature consists, for the local flexisip to send a single INVITE for both alice and bob. The cloud flexisip will be responsible to duplicate and re-target the INVITE to bob and alice
individually. Similarly, the audio and video streams during early media will be factorized: a single video stream will be transmitted from the local flexisip to the cloud one, and the cloud one
will duplicate this stream to bob and alice.
This feature is interesting as the number of users grows, and also because ADSL link of end-users have usually limited upload bandwidth, just good enough for a single video stream but not many.

in [module::Router] section, set this key:

```
allow-target-factorization=true
```

After restarting flexisip, we can see in the logs (--debug option) that a REGISTER is sent to the cloud flexisip.

Smartphone configuration

In order for the smartphone app to be able to receive and send calls through the cloud flexisip, we need to configure a second account.
Go to the assistant, select "use sip account", and enter the following information for this "outdoor account":

Username: bob
password: bobsecret
domain: myhouse
transport: tcp

Again, as "myhouse" is not something known by DNS, registration will fail and we need to enter the account settings in order to set:

Proxy: <sip:mycloudserver.example.org;transport=tcp>
Outbound proxy: YES

Once done, registration is successful.

Testing from outside home

Make sure that the smartphones are outside the home network by for example switching them to mobile data (3G, 4G).

Surveillance call

Sending the call through the cloud server requires that the outdoor account is the active one, that you can do by toggling "use by default" ON in the account configuration.

Type "entry-panel" in the linphone's input field, and press call button. The smartphone will display the video of the entry panel.

Receiving a call

In the same way as in the local test, type "call everyone" in the linphone-daemon shell prompt.
Home screens and smartphone will ring, call can be answered anywhere, and once taken call is canceled for the other devices.

Note: the iOS app will not ring in background mode because of the lack of push notification support in our setup.

Automating configuration of devices and smartphones

For a real world deployment, the configuration of the devices and the smartphones should be done automatically, by means of a configuration management software. For example:

- the smartphone app could have an assistant to let user create their account and securely pair their account with their home network, for example by displaying a QR code in the home screen. The QR code could convey the required information about the home network (domain name, SIP address of local proxy and any kind of additional information). This step should also result in having the appropriate lines added to /etc/flexisip/passwd and /etc/flexisip/routes.conf. Note that instead of a password file, an sqlite database can be used, which is more convenient to add/remove entries from an external program or script.
- The smartphone app should manage by itself the creation of two account configuration (internal, external), using the liblinphone API, and should set the default account to be the cloud one when local registration is not working.

- In order give a fixed hostname to the local flexisip (because IP address may change in the local network), mDNS could be used. In such case -DENABLE_MDNS=ON should be passed to the linphone and flexisip ./prepare.py scripts, because by default the mDNS resolved is not compiled. For convenience, Flexisip has a mDNS section in which it is possible to configure a mDNS name broadcast.
- The configuration management software could also be responsible for providing to smartphone apps the list of entry panels or additional cameras if any.

Security aspects

The setup described in this document is simplified to go fast to the essential: working call flows. But it has negliged most of the security aspects, which are now addressed in this paragraph.

Securing the local flexisip

The following things could be done in order to secure the local flexisip:

- use HA1 encrypted passwords in the /etc/flexisip/passwd file or the database.
- use SIP/TLS instead of SIP/TCP. It requires that the local flexisip has a TLS certificate provided by a certificate authority, which is something that could be handled by the management software at device initialization.
- the domain registration should be done over SIP/TLS, with a client certificate for the cloud flexisip to authenticate the local flexisip, by setting a *tls-certificate-dir* parameter in the domain registration line of domain-registrations.conf

Securing the cloud flexisip

In the current setup, the cloud flexisip does no authentication of clients. It is delagated entirely to the local flexisip, through the accounts setup in the /etc/flexisip/passwd and Authentication module.
A first of level of security could be done by the cloud flexisip in order to allow registrations of authorized smartphone clients and authorized local flexisips.
For that purpose we recommend the use of TLS client certificates in the smartphone app. In details, this means that:

- during user sign-in, the app submits to a web service a CSR (certificate signing request) with CN set to the SIP URI of the user (sip:alice@myhouse)
- the app obtains the certificate and set it together with the private key to the liblinphone, using linphone_core_set_tls_certificate() / linphone_core_set_tls_key().
- when the app will register to the cloud flexisip, liblinphone will automatically present this certificate.

On its side, the cloud flexisip must be configured to use TLS and request client certificates, by setting in [global] section:

```
transports=sips:mycloudserver.example.org;require-peer-certificate=1
```

The authentication module shall be enabled:

```
[module::Authentication]
enabled=true
auth-domains=*
```

The cloud flexisip will then automatically reject with 401 or 407 any request whose *From* header doesn't match the CN claimed
in the certificate presented during the TLS handshake of the connection on which the request is received.

Securing audio and video

Now that signaling is secured with SIP/TLS, we can enable ciphering of audio and video packets (RTP/RTCP). This can simply be done by using linphone_core_set_media_encryption() with LinphoneMediaEncryptionSRTP.

Push notifications

On iOS, an app cannot receive any network packet when running in background, or when the screen is off. This is constraint from Apple to save battery. Many Android devices don't suffer from this limitation, but it tends to be more and more rare. As a result, enabling push notifications in the cloud flexisip and the apps is an important requirement, in order to make sure that incoming calls are delivered to the smartphones, even when the smartphone is at home.

It is somewhat complex from a security standpoint to enable push notifications in the local flexisip, because it means that the entry panel device will contain the certificates for sending push notifications to the apps. However it is not necessary to do this, as the cloud flexisip can send push notifications even in the case where the smartphone is in the house.
This is achievable by configuring the smartphone app to always keep a registration to the cloud flexisip, with a long registration expire time (for example: 12 months), so that the cloud flexisip always has up to to date push notification tokens of all devices in its registration database.
Even when the smartphone is in the home network, the local flexisip will always transmit an INVITE to the cloud flexisip, and then the cloud flexisip will take this opportunity to send a push notification to the smartphone app. The app will then wake up and immediately attempt to register with both local and cloud SIP accounts. It is expected that local registration will happen faster than the cloud one, and hence the INVITE will be received from the local flexisip. Liblinphone will decline with 486 Busy the INVITE sent by the cloud flexisip, as it will notice that it is a duplicate.

Specific documentation about push notification enablements in flexisip and apps is available on this page.

Conclusion

This document explains a typical design elaborated by Belledonne Communications to reach video call flows between a local network at home and public internet, using Linphone and Flexisip products. This is not the only possible solution and some variants for various aspects do exists. Feel free to consult us in order to elaborate the best solution for your needs !