Telive - Tetra live receiver v0.7

(c) 2014 Jacek Lipkowski <sq5bpf@lipkowski.org>

Description

Telive is a program which can be used to display information like signalling, calls etc from a Tetra network. It is also possible to log the signalling information, listen to the audio in realtime and to record the audio. Playing the audio and recompressing it into ogg is done via external scripts. The software is based upon slightly modified osmocom-tetra software.

Prerequisites

Before installing please read all licenses, disclaimers, documentation for all installed packages, and about the Tetra protocol (i will not explain term like MNC, Colour Code, Usage identifier, SSI etc).

The software was tested on Debian GNU/Linux 6.0 and 7.0 64 bit versions (with the codec compiled in a 32-bit environment). Other linux distributions should work, but i have not tested them. It should probably be possible to run this software on other systems, but i haven't tried, and don't intend to (however if it works for you, please send me a detailed description so that i can add it to the docs).

Receiving is done using a RTL2832/R820T DVB-T USB dongle. Probably other receivers supported by gnuradio will work with minor modifications.

The following software needs to be installed:

Gnuradio 3.6 (not 3.7!) with osmosdr support.
Probably the easiest way is to use the build-gnuradio script provided by sbrac:

http://www.sbrac.org/files/build-gnuradio

Please run ./build-gnuradio -o (the -o parameter installs version 3.6).

After building, test if the software works. There are many gnuradio-companion scripts on the internet, see if they work, and also use them to find the right ppm value for your rtl-sdr dongle.

libosmocore-sq5bpf

This is Osmocom libosmocore, the original software is here:
http://bb.osmocom.org/trac/wiki/libosmocore
Please read all documentation for this project.
The version in my repository is not patched in any way, but may be patched in the future.

To compile and install:
git clone https://github.com/sq5bpf/libosmocore-sq5bpf
cd libosmocore-sq5bpf
autoreconf -i
./configure
make
sudo make install

osmo-tetra-sq5bpf

This is a patched version of Osmocom osmo-tetra, the original software is here: http://tetra.osmocom.org/trac/wiki/osmo-tetra

To compile:
git clone https://github.com/sq5bpf/osmo-tetra-sq5bpf
cd osmo-tetra-sq5bpf
cd src
make

The scripts receiver1 and receiver2 assume that you are in this directory (osmo-tetra-sq5bpf/src).

Tetra codecs

Please read the instructions in osmo-tetra-sq5bpf/etsi_codec-patches , including README_sq5bpf. This shows how to obatin and compile the codecs. Warning: the codecs won't run when compiled for 64-bits, please either use a compiler that produces 32-bit binaries, or set up a 32-bit build system in chroot().

telive

To compile:
git clone https://github.com/sq5bpf/telive
cd telive
make
sudo mkdir /tetra
sudo chown YOURUSER.YOURGROUP /tetra
./install.bin

The scripts rxx and rxx2 assume that you are in this directory.

Optional step

Get the latest wireshark from the repository. Wireshark can parse GSMTAP messages which are sent via tetra-rx and display the decoded Tetra frames.

Theory of operation

Please refer to Fig 2. The RF signal is received via na rtl-sdr dongle and gnuradio (the receiver is conviniently provided as an gnuradio-companion flow graph to aid easy modification). The received channel is passed via a named pipe /tmp/fifo1 to a CQPSK demodulator simdemod2.py. The demodulator output is fed to float_to_bits ("bit slicer"), and further to tetra-rx (which decodes the tetra protocol). tetra-rx sends some packets encapsulated in UDP to localhost. These packets are received with the telive program.

The telive program shows a list of possible usage identifiers (0-63, where 0-2 are reserved), and aggregates signalling information (SSI addresses etc) for each usage identifier. If there are any voice frames, then this information can be recorded or played back immediately. Later the recording is renamed to a filename containing the date, time and the last 3 SSI numbers seen for this usage identifier.

The tetra-rx program has been modified to receive SDS messages, and these messages can be logged by the telive program.
The telive program has these settings:

mutessi - allow recording and playback of data for a usage identifier with no SSI data (useful, because it supressed the playback of encrypted data)
alldump - don't filter signalling information, show all in the log and in the message window
mute - mute playback audio (but not the recording)
record - record audio
log - log signalling information to file (default telive.log)

Fig 1. telive program screen

Keyboard commands:
? - show help in the status window
m - toggle *mutessi*
M - toggle *mute*
r - refresh screen (warning: segfaults sometimes, to be fixed)
R - toggle *record*
a - toggle *alldump*
l - toggle logging
s - stop play (if there are multiple channels active, this will end the active playback and search for another channel to play).

The numbers 0-63 are the usage identifiers. If you see OK near the number then there is voice traffic present on it. If you see PLAY then this is the currently playing channel.

There can be multiple tetra-rx programs feeding data to one telive process, but be certain that they are on the same tetra network (same uplink/downlink, same colour code etc) - refer to Fig 3 for an example. It is also possible to feed totally different channels (from one or more gnuradio instance), each to its own named pipe, have many processes to decode them, and feed multiple

telive processes from the receiver processes.

Telive records the calls in in ACELP codec format. You can use the script tplay to play them, or tetrad to automatically encode them into OGG format.

Simple 1 channel setup example:

This is a simple receiver for one frequency only (it is best to use the control channel frequency, as it contains the most signalling information).

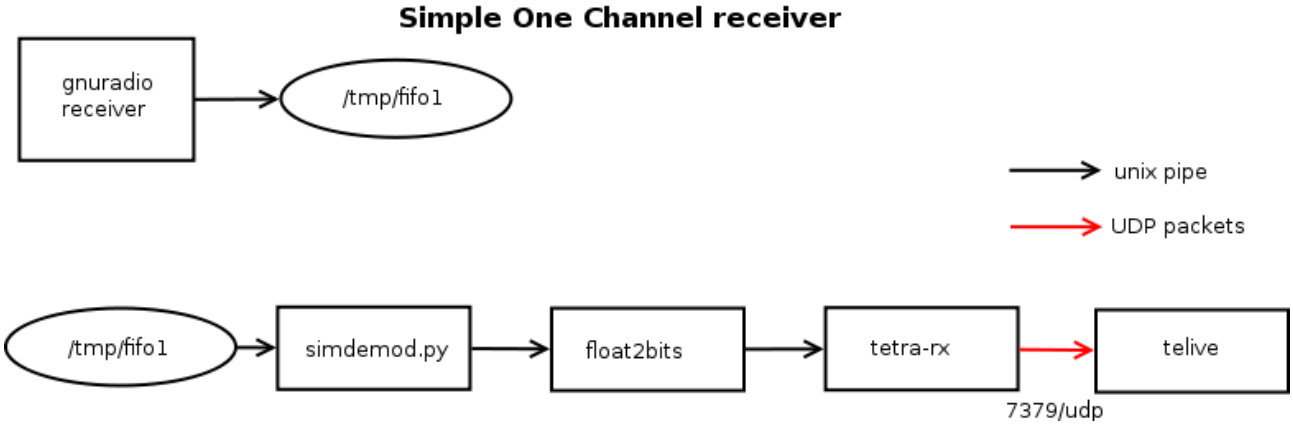

Fig 2. Simple one channel receiver

Open an xterm, change directory to the osmo-tetra-sq5bpf/src directory and execute
./receiver1 1
There should be no errors. This creates /tmp/fifo1 , and listens for incoming data, piping it into simdemod.py, float_to_bits and tetra-rx

Open another xterm with:
/usr/bin/xterm -font fixed -bg black -fg white -geometry 203x60
Change to the telive directory and execute:
./rxx

This launches the telive console. In the telive console pressing shift-R enables recording of voice calls, and pressing L enables logging. If you hear too much gibberish press m (enables mutessi).

In another xterm launch /tetra/bin/tetrad - this will recode voice in the ACELP format into OGG format, and put it into /tetra/out

Connect a rtl-sdr dongle and antenna. Open telive_1ch.grc (from the telive/gnuradio-companion directory) in gnuradio-companion, and run it. Currently telive_1ch.grc defaults to 435.500MHz and 56ppm. Please adjust the tuner main frequency, offset and ppm to receive a known strong tetra signal (preferably unencrypted). Correct the ppm value so that the spectrum looks "symmetrical".

If all is set up correctly, the xterm where receiver1 is run should scroll a lot of text, while the telive console should display the MCC, MNC, Colour Code and uplink/downlink frequencies for the control channel. If there is any traffic you should see some SSI numbers on the console, and maybe hear some voice (voice is muted with the key command shift-M).

Two channel setup example

This is a receiver for two frequencies, preferably one signalling channel, and some other channel from the same network The setup is similar to Fig. 2, but there are two receiver1 processes, and gnuradio receives two channels simultaneusly. Please refer to Fig 3

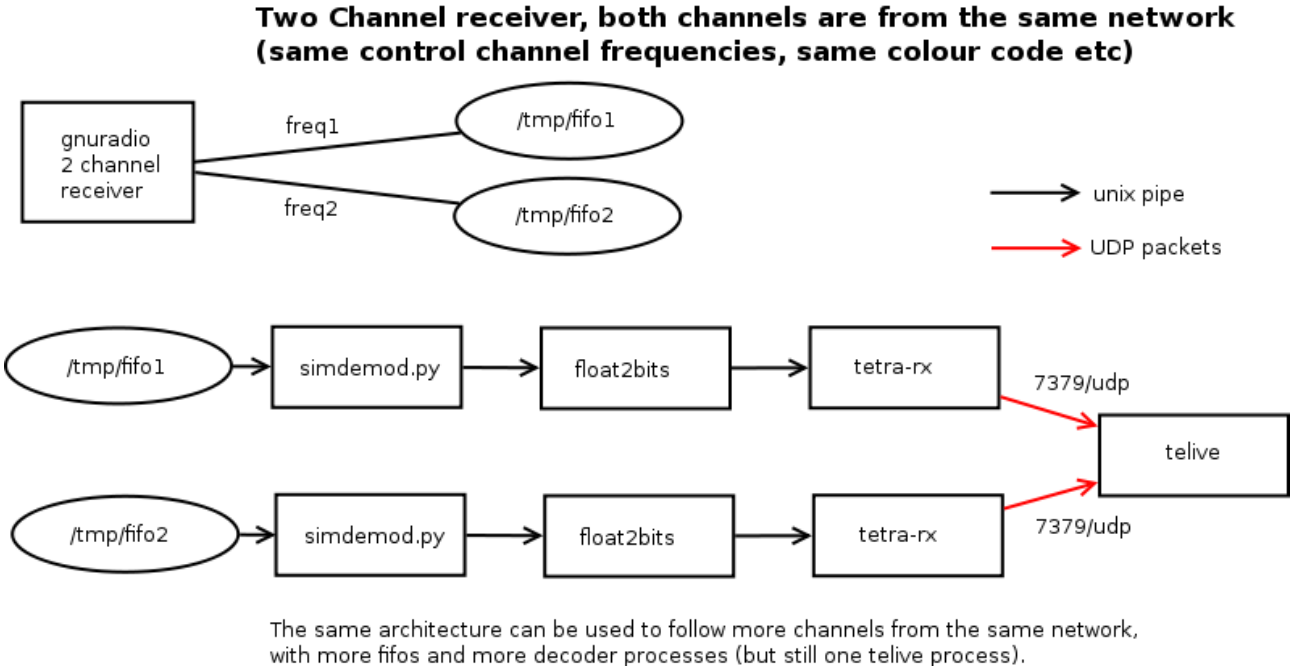

Fig 3. Two channel receiver setup

Open an xterm, change directory to the osmo-tetra-sq5bpf/src directory and execute
./receiver1 1
There should be no errors. This creates /tmp/fifo1 , and listens for incoming data, piping it into simdemod2.py, float_to_bits and tetra-rx

Open an xterm, change directory to the osmo-tetra-sq5bpf/src directory and execute
./receiver1 2
There should be no errors. This creates /tmp/fifo2 , and listens for incoming data, piping it into simdemod2.py, float_to_bits and tetra-rx

Open another xterm with:
/usr/bin/xterm -font fixed -bg black -fg white -geometry 203x60
Change to the telive directory and execute:
./rxx

Connect a rtl-sdr dongle and antenna. Open telive_2ch.grc (from the telive/gnuradio-companion directory) in gnuradio-companion, and run it. Currently telive_2ch.grc defaults to 435.500MHz, 434.750MHz and 56ppm. Please adjust the tuner main frequency, offsets and ppm to receive a known strong tetra signal (preferably unencrypted). Correct the ppm value so that the spectrum looks "symmetrical"..

Please note that you will get more data then with the 1 channel receiver setup. If the tetra network attributes change in telive, then the two channels are not from the same network, and this won't work correctly.

Four channel, two networks setup

This is a receiver for four frequencies, to monitor two separate Tetra networks, each with two channels. This is basically the twice the Two channel setup from Fig. 3 .

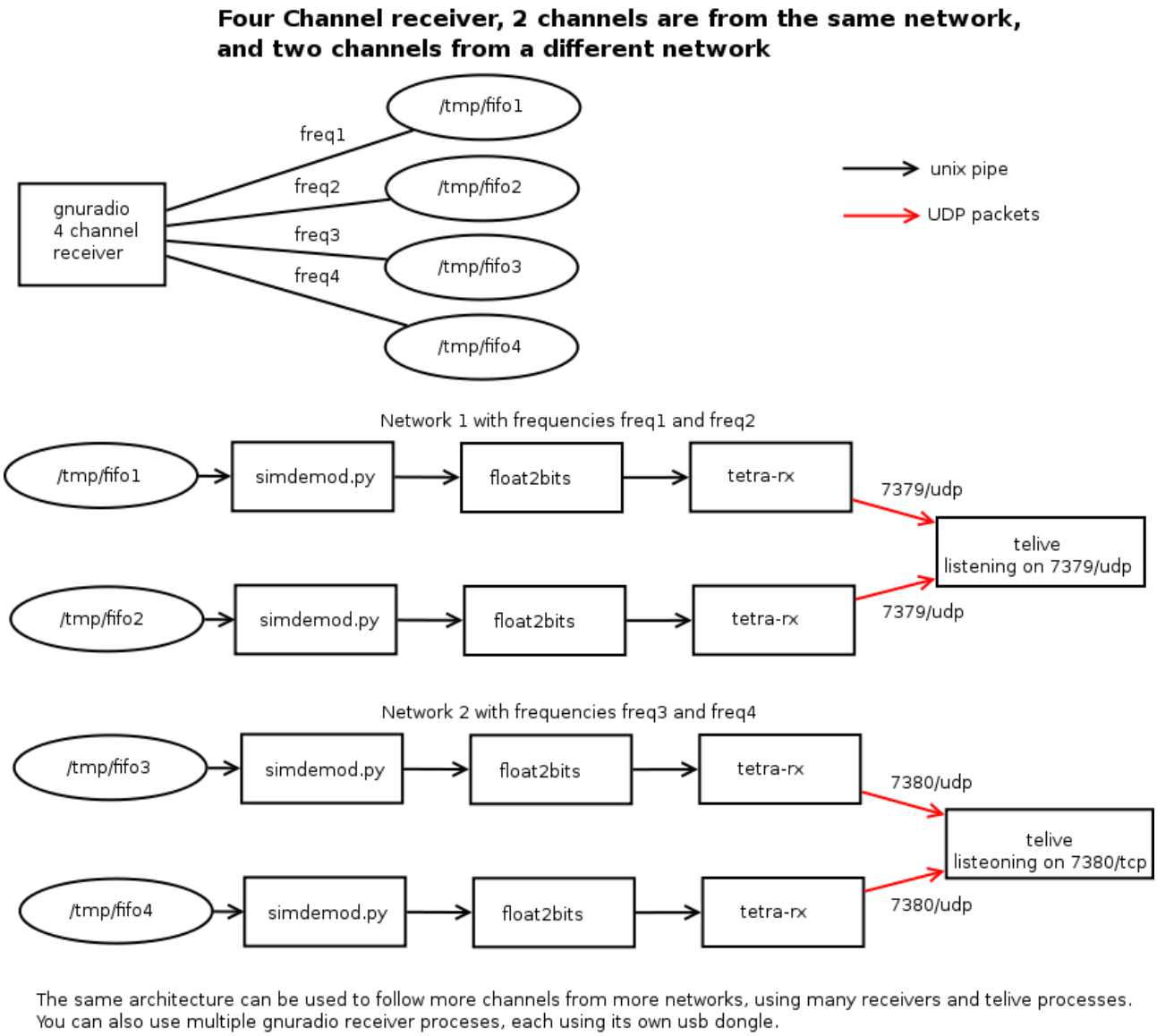

Fig 4. Four channel, two networks setup

Network 1:

Open an xterm, change directory to the osmo-tetra-sq5bpf/src directory and execute
./receiver1 1
There should be no errors. This creates /tmp/fifo1 , and listens for incoming data, piping it into simdemod2.py, float_to_bits and tetra-rx . This process will send data via UDP to 7379/udp.

Open an xterm, change directory to the osmo-tetra-sq5bpf/src directory and execute
./receiver1 2
There should be no errors. This creates /tmp/fifo2 , and listens for incoming data, piping it into simdemod2.py, float_to_bits and tetra-rx . This process will send data via UDP to 7379/udp.

Open another xterm with:
/usr/bin/xterm -font fixed -bg black -fg white -geometry 203x60
Change to the telive directory and execute:
./rxx
This telive process will listen to 7379/udp

Network 2:

Create a directory /tetra2 analogous to /tetra.

Open an xterm, change directory to the osmo-tetra-sq5bpf/src directory and execute
./receiver2 3
There should be no errors. This creates /tmp/fifo3 , and listens for incoming data, piping it into simdemod2.py, float_to_bits and tetra-rx . . This process will send data via UDP to 7380/udp.

Open an xterm, change directory to the osmo-tetra-sq5bpf/src directory and execute
./receiver2 4
There should be no errors. This creates /tmp/fifo4 , and listens for incoming data, piping it into simdemod2.py, float_to_bits and tetra-rx . . This process will send data via UDP to 7380/udp.

Open another xterm with:
/usr/bin/xterm -font fixed -bg black -fg white -geometry 203x60
Change to the telive directory and execute:
./rxx2
This telive process will listen to 7380/udp

Connect a rtl-sdr dongle and antenna. Open telive_4ch.grc (from the telive/gnuradio-companion directory) in gnuradio-companion, and run it. Currently telive_4ch.grc defaults to 435.500MHz, 434.750MHz, 435.750MHz, 435.800MHz and 56ppm. Please adjust the tuner main frequency, offsets and ppm to receive a known strong tetra signal (preferably unencrypted). Correct the ppm value so that the spectrum looks "symmetrical".

Other setups

Please look at the scripts and the program sources. You can use multiple gnuradio receivers each with its own rtl-sdr dongle, and feed the results via named pipes to multiple receiver processes. The results can be displayed via multiple telive processes, each for one Tetra network.

End notes

This is code that i've written for my own pleasure. It is very ugly, but i've could either polish it so it looks nice, or "release early, release often" (per the open source methodology). Polishing it would take forever, and i believe that if one has a nice toy, he should share it with other children, rather then keep it to himself.

This code runs under linux, and was tested under Debian 6.0 and 7.0 64-bit versions. I will not answer questions how to make it work under opertating systems different than linux, however if you make it work, send me a description, so that i can put it into the documentation. Also i haven't really tried to make this user-friendly. The documentation is crap, despite my best efforts to write it. Basically read all of the documentation, the ETSI Tetra docs, and if you still don't understand something, then RTFS.

The architecture is modular, and you could probably use another receiver instead of gnuradio (like rtl_fm). This hasn't been tried, but should work, and should result in less CPU utilisation.
The telive program uses the usage identifier as the key. This concept works, but is not correct. A more correct approach would be to use the notification id and receiver id. This will probably be implemented in some later version. The SDS decoding might not work in many cases. So far i've seen only type 0x82 (Text messaging), 8 bit encoding type SDS. 7-bit decoding might not work etc. I would also like to implement Location services, and data transfer services.

If you can provide samples of interesting traffic legally, then please do. It would be best if you have your own TMO Tetra network, that you could monitor, while changing various settings etc.

The core of this software: osmocom-tetra, libosmocore was written by Osmocom (i've only filled in some blanks, that others probably didn't want to fill in). As stated on their web page http://tetra.osmocom.org/trac/wiki/FAQ "Can I use OsmocomTETRA to listen to police radio? [...] Also, if this is your interest in this project: Please simply go away, we don't want to talk to you. " - please respect this.

Note about Tetra security

OMG! Someone can listen to my secret tetra tranmissions! The sky is falling! Well, actually not.

All these digital systems (Tetra, DMR, NXDN etc) have an option to either encrypt traffic or not. This is a major advantage over analog systems, where it was hard to encrypt. Now if a system doesn't have encryption enabled, than one can assume that it is a conscious choice of the system designer, and that the system is intended to be open to monitoring.

The encryption algorithms are not public, so it is hard to tell if there are any flaws in them. Probably sooner or later someone will reverse engineer or leak them (as was with GSM A5/1 and other algorithms).

Disclaimer

I disclaim any liability because of the use of this software. If someone breaks something it's their fault. Also please observe the licenses. The telive software is licensed GPLv3. The osmosom-tetra software is licensed GNU Affero v3, and libosmocore is licensed GPLv2. The codecs are licensed by ETSI, and if i understand correctly, you can build binaries for yourself from the publically avaliable source code, but might not be able to provide the binaries to others. IANAL, so don't take this as legal advice, if someone knows better, then please correct me on this.

TODO

Implement a better protocol to communicate with telive
Implement Location Information Protocol and Data Services
Don't use popen() for playback because of buffering.
Actually try to read the ETSI docs and not fall asleep after 20 pages.
Clean up the code, rewrite all comments in english (if there are any polish comments or variable names left).
Write proper documentation