Telive - Tetra live receiver v1.5

(c) 2014-2015 Jacek Lipkowski <sq5bpf@lipkowski.org>

Description

Telive is a program which can be used to display information like signaling, calls, SDS, location information etc from a TMO 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 re-compressing it into ogg is done via external scripts. The software is based upon slightly modified osmocom-tetra software. Location information can be written to a KML file, to be opened in Google Earth and similar 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). Read this whole document, preferably multiple times.

The software was tested on Debian GNU/Linux 6.0, 7.0 and 8.0 64 bit versions. 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). The software is compatible with gnuradio 3.6 and 3.7 (newer versions, 3.7.2 doesn't work, 3.7.5 does), with the osmosdr source block.

Receiving is done using a RTL2832/R820T DVB-T USB dongle. Probably other receivers supported by gnuradio will work with minor modifications (people have successfully run it with the funcube dongle, hackrf and other hardware).

To simplify things, a install_debian8.sh script is provided to build and install everything under Debian 8 (see the FAQ section: How do I install under Debian 8), use this unless you have a valid reason to install manually.

Manual Install:

Getting GNURadio

A supported gnuradio version can be installed either from the distribution packages (gnuradio 3.6 on Kali Linux and gnuradio 3.7.5 on Debian 8 are known to work), by building it with pybombs (not covered here, but described on many web pages), or by building it with the SBRAC build-gnuradio script (supports both versions 3.6 and 3.7, please see the “How do I install gnuradio using the build-gnuradio script” section in the FAQ). Use gnuradio 3.7.x (where x>=5), unless there is a valid reason to use gnuradio 3.6 (such as other software which requires version 3.6).

Installing the patched Osmo-tetra, codecs, telive et al

This is moved to the FAQ section “How do I manually install the osmo-tetra-sq5bpf, codecs, telive”.

Optional step

If your network provides location information, then install Google Earth.

Get the latest wireshark, preferably the latest version from the repository (the packages on recent distributions may work too). Wireshark can parse GSMTAP messages which are sent via tetra-rx and display the decoded Tetra frames.

Install some convenient music player to play the recorded ogg files. Audacious is one example of such player.

Theory of operation

Please refer to Fig 2. The RF signal is received via na rtl-sdr dongle and gnuradio (the receiver is conveniently 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 signaling 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, displayed on the top line of the screen:

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 signaling 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)
verbose – verbosity level (more shows more debug info)
lock – shows if this telive instance is prohibited from playing by another telive instance

Fig 1. telive program screen

Keyboard commands:

? - show keystroke 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).

V/v – increase/decrease verbosity

f – toggle SSI filter disabled/enabled/inverted

F – enter SSI filter expression

t – switches between main window/frequency window (and other implemented in the future)

z – forgets any information that was learned about this network

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, same LA 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.

The message window shows tetra signaling messages (SETUP, RELEASE etc). The status window will show various information regarding program status (depending on the verbosity level), and text SDS messages.

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.

Telive has the ability to filter the usage identifiers which are played live based upon SSIs. The filter expression is the same as shell wildcards in the ksh shell (see the FAQ for some examples). The expression can either be passed in an environment variable, or you can press F (capital f) and enter the expression. The filtering can either be disabled, enabled (only conversations with SSIs matching the filter are played live), or inverted (only conversations with SSIs NOT matching the filter are played live). The filtering mode is toggled with f (small f).

Telive can periodically log location information into a KML file (see the FAQ for some examples). Currently LIP Short Location Report, and a proprietary Simple Location System User Application 0x80 used in Spain (not sure what company is behind this). KML files can be opened in software like Google Earth.

Pressing z will toggle windows, showing either the usage identifiers or a list of frequencies that the network broadcasts in various messages.

Note: the following examples were written for gnuradio 3.6, for version 3.7 use the flowgraphs with _gr37 in the filename (for example telive_1ch_simple_gr37.grc).

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 signaling information).

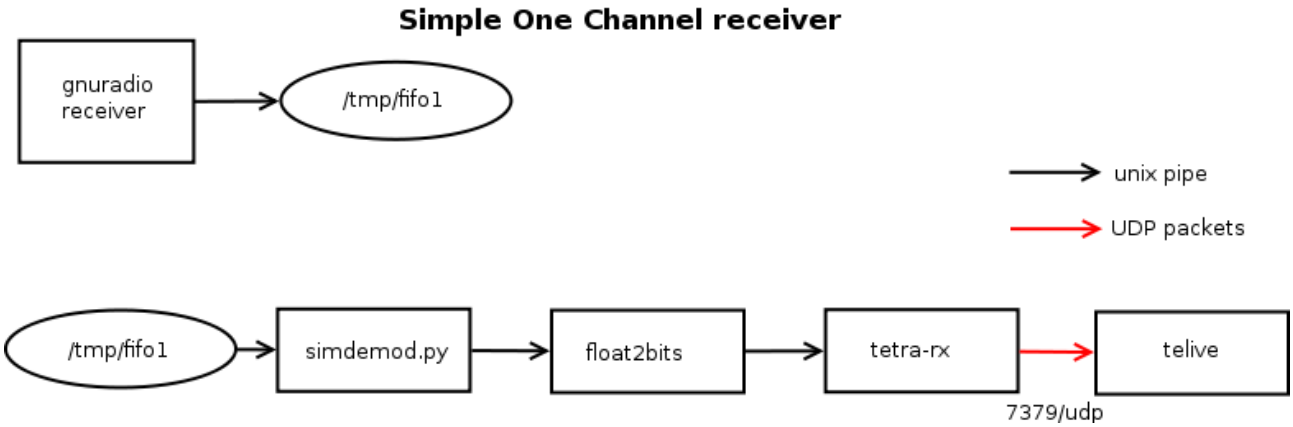

Fig 2. Simple one channel receiver

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 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

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". Instead of telive_1ch.grc, telive_1ch_simple.grc can be used, this is more user-friendly, shows spectrum, has click-to-tune etc (at the expense of a bit more cpu utilisation).

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 signaling 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 simultaneously. Please refer to Fig 3

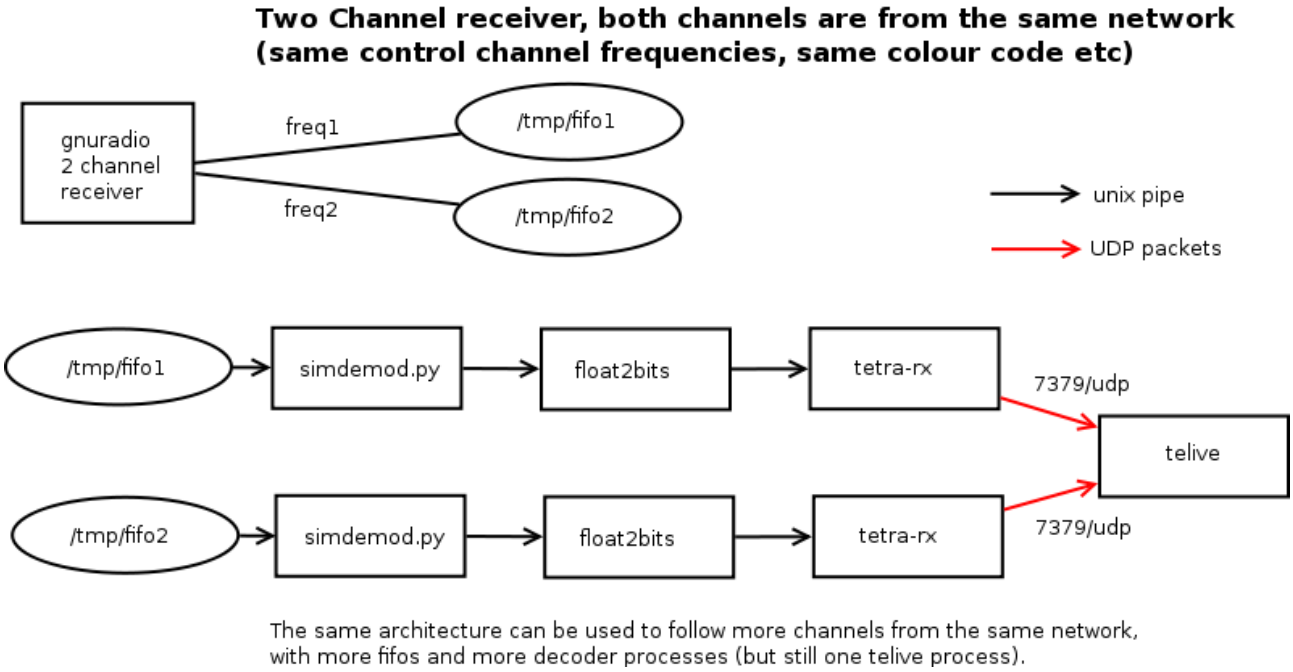

Fig 3. Two channel receiver setup

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 .

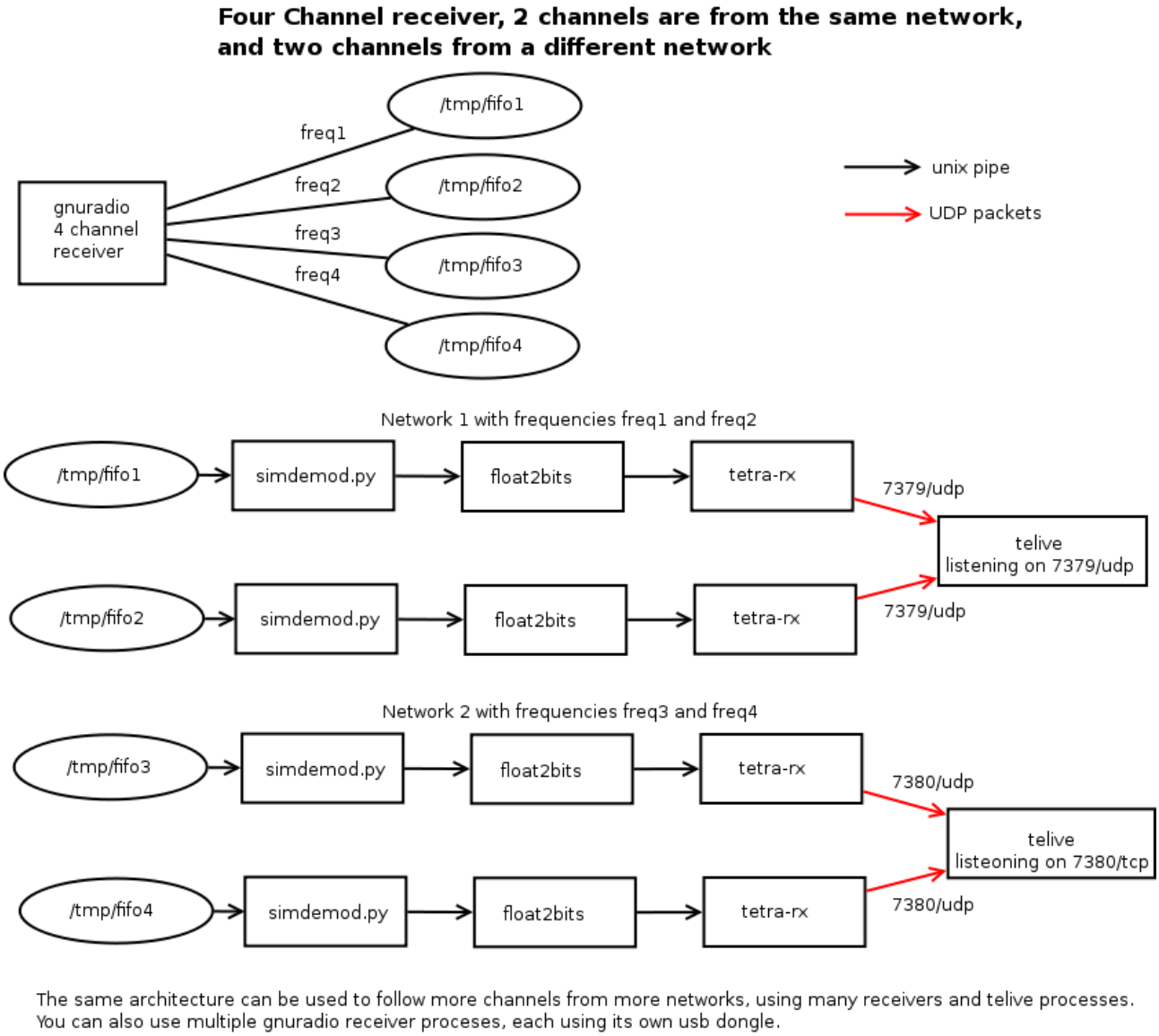

Fig 4. Four channel, two networks setup

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
In the new xterm window that opened:
Check with stty -a that there are indeed 60 rows and 203 columns.
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. Setting up the various paths is left as an exercise to the reader.

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".

If there is voice traffic in both networks, then both telive instances will play at the same time. The locking feature can be used to make only one instance play at a time. To do so please uncomment the export TETRA_LOCK_FILE=/tetra/telive_lock line in rxx and rxx2.

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.

List of gnuradio-companion flowgraphs

There are several flowgraphs in telive/gnuradio-companion:
telive_1ch.grc, telive 2ch.grc … upto telive_6ch.grc – 1,2 … 6 channel receivers
telive_1ch_simple.grc – this receiver doesn't require setting of the offset, presents also the full band spectrum, and has click-to-tune capability (and should be generally more user-friendly)
The files with _gr37 in the name are for gnuradio 3.7.

End notes

This is code that i've written for my own pleasure. It is very ugly, but i 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 6.0,7.0 and 8.0 64-bit versions. I will not answer questions how to make it work under operating 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 and some software to recode the output to a supported format). This hasn't been tried, but should work, and should result in less CPU utilization.
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. Many protocols used over SDS are proprietary, but I will try to support them if any info is avaliable.

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.

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
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

FAQ

What does this software do?

It enables one to listen and record unencrypted TETRA calls, decode some SDS messages and log signalling information.

Where can I learn more about TETRA?

The TETRA standard is avaliable from ETSI, go to http://etsi.org and search for TETRA. You will get more documents than you will ever want to read. Please note that TETRAPOL is something completely different (the only similarity being the first 5 letters :).

What operating systems does it run on?

It runs on linux. It was developed under Debian 6.0, 7.0 and 8.0 (all 64-bit). Others have reported that it works under Kali linux, Mint and Ubuntu. Any Unix/Linux operating system should work with some tweaking (preferably the debian-based ones), if you can get a supported gnuradio version working on it.

Does it work on Microsoft windows?

No. And i'm not interested in it. However you can run it in a virtual machine, some have reported success when running under vmware player, and under virtualbox. Beware that a virtual machine doesn't run as fast as a normal installation, and might not be suitable for more advanced setups (monitoring multiple channels etc). Also there are issues regarding access to USB devices from virtual machines.

I don't understand all this, could you provide a nice description how to get it to work for people who don't know much about computers or radio?

Not really. Go away. And please don't send emails asking for this. Thanks!

If Linux is a problem, then go find some friendly linux user with at least basic skills (ie. one that can do something more then point the cursor at icons and press a button).

What is a usage identifier (also called usage marker)?

From the tetra docs: “A traffic usage marker is a 6-bit MAC label used during circuit mode calls for transmitter preemption, for prevention of crossed calls and for channel maintenance purposes. The BS shall assign a traffic usage marker before any traffic transmission takes place on an assigned channel.”

This identifier is present during various call setup procedures, and also accompanies voice frames. This identifier is used by telive to bind voice traffic with signaling information. The identifiers 0-3 are reserved, and you should not see them in use in telive.

How can I record calls?

Press shift-R (the top line should read record:1). The calls are recorded in ACELP format in the directory /tetra/in. The script tetrad will recompress them into OGG files and put them in:
/tetra/out/YYYYMMDD/traffic_YYYYMMDD_HHMMSS_UU_SSI1_SSI2_SSI3.ogg
YYYYMMDD is year month date (like 20141127)
HHMMSS is hour minute second (like 230145)
UU is the usage identifier
SSI1, SSI2, SSI3 are the last 3 SSI numbers associated with this usage identifier

How can I log signalling?

Press L (the top line should read log:1). The log is in telive.log (this can be changed by setting the environment variable TETRA_LOGFILE). This log contains the signalling information in a readable form (not as long as the tetra-rx output), and SDS messages (the text messages should be decoded).
The most information is present in the tetra-rx output. To log it do:
./receiver1 1 2>&1 |tee -a /tmp/logfile
Beware that this will eat away many megabytes of disk space per minute.

I don't hear anything or something stutters, but the recordings are fine if I play them on another system

This has nothing to do with telive. Find some wave file and play it using aplay. Fiddle with the system until it's fixed (maybe change the mixer settings? See support forums for your distribution for possible fixes).

Stuttering is often caused by not enough cpu power avaliable, often when monitoring multiple channels, or when running under a virtual machine (which is not recommended also for this reason). You can lower the cpu load by:

- stopping the spectrum display
- monitor less channels
- don't run additional software (especially Google Earth), or run it on another computer
- using a lower sample rate for the rtl-sdr dongle (search the internet for supported rates for your receiver)
- check if your computer is running at full speed, disable laptop-mode, look at cpufreq etc

The telive program exits when it sees voice traffic. Why?

There is a problem running /tetra/bin/tplay , or a problem running sdecoder/cdecoder/aplay. Note: as of version 0.9 telive should no longer exit, but print an error message.

How can I check if the codecs are compiled ok, and if audio works?

testfile.acelp contains a compressed voice sample. Please try:
/tetra/bin/tplay < testfile.acelp
You should hear “Hello Tetra”, if not debug the tplay script, sdecoder/cdecoder/aplay. Also look at the mixer settings, maybe the audio has been muted?

Something segfaults, how can I help debug it?

Please email me the result of these commands:

gdb program_that_segfaulted core
bt

The usage identifiers at the bottom are all on the same line, like 9:10:11:12: etc

This means that the xterm screen size is not 203x60 (maybe your window manager resized it). The software should still work, but the screen will be a bit unreadable.

I get fseek failed from receiver1

This means that something has closed /tmp/fifo... Most probably the gnuradio-companion receiver has died. Check for errors in the gnuradio-companion console.

I hear mostly unreadable audio

This is probably audio from encrypted calls, or some non-voice data. Enable mutessi – this will ignore usage identifiers which don't have at least one SSI number assigned.

How do I use filters?

To use a filter you have to enter a filter expression, either by pressing F and entering a filter expression, or by passing the expression in the TETRA_SSI_FILTER environment variable. The filter expressions are ksh extended filename matching expressions (for a better explanation please search for shell wildcard expresions, or for fnmatch and FNM_EXTMATCH).

? wildcard that matches one character
* wildcard that matches any amount of characters (don't use this for SSIs)
[1-5] matches 1,2,3,4,5
[2389] matches 2,3,8,9
[this is from https://ftp.gnu.org/old-gnu/Manuals/glibc-2.2.3/html_chapter/libc_10.html]
The patterns are written in the form explained in the following table where *pattern-list* is a | separated list of patterns.

`?(`*`pattern-list`*`)`
: The pattern matches if zero or one occurences of any of the patterns in the *pattern-list* allow matching the input string.

`*(`*`pattern-list`*`)`
: The pattern matches if zero or more occurences of any of the patterns in the *pattern-list* allow matching the input string.

`+(`*`pattern-list`*`)`
: The pattern matches if one or more occurences of any of the patterns in the *pattern-list* allow matching the input string.

`@(`*`pattern-list`*`)`
: The pattern matches if exactly one occurence of any of the patterns in the *pattern-list* allows matching the input string.

`!(`*`pattern-list`*`)`
: The pattern matches if the input string cannot be matched with any of the patterns in the *pattern-list*.

Examples:

1000 – match SSI 1000
10?? - match SSI 1000-1099
+(1000|[234]0??|?????) - extended pattern, matches 1000, 2000-2099, 3000-3099, 4000-4099, and any 5 digit SSIs

Can I have the software start with some defined state, so that I don't have to change any settings when it starts?

For telive look at the environment variables that it uses. For example in rxx you can put:
export TETRA_KEYS=RMl
This will work exactly the same as if the keys R (record), M (mute) and l (logging) were pressed.

For osmo-tetra-sq5bpf look at the receiver1 script, the UDP port number and receiver number can be set there.

For the gnuradio-companion flowgraphs it is best to make a copy of the flowgraph, load it into gnuradio-companion, edit the variables (like ppm, frequency, offset etc), and save it again. The flowgraph can also be compiled into a python script, that can be launched directly without gnuradio-companion (click Build->Generate, a file with a .py extension will appear in the same directory as the grc flowgraph).

Can I have nice textual descriptions for SSIs?

The file ssi_descriptions contains lines like:
SSI_number<exactly one space>SSI_description
Please strictly adhere to this format, no empty lines etc.

The supplied ssi_descriptions file has a few examples.

The filename can be overridden with the TETRA_SSI_DESCRIPTIONS environment variable.

What environment variables are used by telive?

TETRA_OUTDIR – the directory where telive records voice calls. If unset /tetra/in is used
TETRA_LOGFILE – the file that signalling information is logged to. If unset telive.log is used
TETRA_PORT – the udp port which is used for communication with tetra-rx. If unset 7379 is used
TETRA_SSI_FILTER – the SSI filtering expression
TETRA_SSI_DESCRIPTIONS – name of the file containing textual descriptions of SSIs. If unset ssi_descriptions is used
TETRA_KEYS – contains a list of characters that will be parsed by telive, just as keystrokes would (so for example if you set it to Rff it will enable record and inverted SSI filter), don't use this for inputting the filter expression (use TETRA_SSI_FILTER for this)
TETRA_KML_FILE - if set, the locations will be written periodically to this file in KML format
TETRA_KML_INTERVAL - this will set the maximum KML file refresh rate. If unset, this will default to 30 seconds
TETRA_LOCK_FILE – if this is set, then this filename will be used for locking between telive instances
Please look at the rxx script where some of these variables are set.

How can I access location information?

The software understands LIP SHORT LOCATION REPORT, and one proprietary Simple Location System coding scheme (more will be added later, please send me samples of telive.log, and your approximate longitude / latitude). If a filename is provided in the TETRA_KML_FILE environment variable, then telive will write location information to this file periodically (uncomment the line that sets this variable for a demo). This file can be opened via Google Earth, or any other software supporting KML.

For near realtime tracking the KML file can be referenced as a Network Link (refer to the Google Earth documentation for further information). A demo is provided in the example_google_earth.kml file – it will reload /tetra/log/tetra1.kml every 5 seconds. It is also possible to serve the KML file via HTTP to multiple remote machines running Google Earth, and modify example_google_earth.kml to reference the file via a http url.

The maximum refresh frequency (in seconds) is set via the environment variable TETRA_KML_INTERVAL. Writing the KML file is done in the same process as the rest of telive, and thus can block the rest of telive for a moment. If you have a lot of location information, and hear a lot of stuttering when KML writing is enabled, then increase TETRA_KML_INTERVAL.

Not much location protocols are implemented at this moment. Please provide samples of telive.log with unimplemented protocol dumps, and with your approximate longitude / latitude. This will help implement them.

Can I use a different receiver?

You can use any receiver that will be able to write baseband data to a pipe in the same format as gnuradio does.

How do I automagically install everything under Debian 8

There is an experimental script to install everything under Debian 8. It will compile/install gnuradio, osmo-tetra-sq5bpf, codecs and telive. The

Steps to install:

- install Debian 8, preferably 64-bit
- ensure that you have internet access.
- setup sudo, with privileges for your regular user (search the internet if you don't know how to do this)
- check if sudo works, for example like this:

```
sudo id
[sudo] password for YOURUSER:
uid=0(root) gid=0(root) groups=0(root)
```

- as your regular user:

```
wget https://raw.githubusercontent.com/sq5bpf/telive/master/scripts/install_debian8.sh
chmod 755 install_debian8.sh
# now here you should read the script, don't just blindly run scripts off the internet
./install_debian8.sh
```

- read the script output, look if there are any errors etc

The whole installation should take only a few minutes.

Adventurous users can try to use this under other linux distributions: first install gnuradio 3.7.5 (or higher), comment out the version check in the script, and run the script. Doing this is left as an exercise to the reader :)

How do I install gnuradio using the build-gnuradio script

Probably the easiest way is to use the build-gnuradio script provided by SBRAC. It supports OpenSuSE, Debian, Ubuntu, Mint, Fedora, Redhat.

Make sure you have sudo privileges.
wget http://www.sbrac.org/files/build-gnuradio
chmod 755 build-gnuradio

To install gnuradio 3.7 please run ./build-gnuradio
To install gnuradio 3.6 please run ./build-gnuradio -o

New installations should use gnuradio 3.7. The gnuradio 3.6 support was known to be broken a few times in this script, and it should be only used when 3.6 compatibility is needed for some other software.

This script is widely used by many people, if you run into a bug, then search the internet for answers (and don't bug me, as i'm not the author).

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.

How do I manually install the osmo-tetra-sq5bpf, codecs, telive

Please install the following software from your distribution: oggenc (package vorbis-tools under debian), sox (package sox), aplay (package alsa-utils), ncurses development libraries (package libncurses-dev). You will need all of the development packages (make, gcc, git etc, under debian most of this is in build-essential). The following assumes that you have sudo privileges.

- 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. It is there only to insure that the version doesn't change.
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).

- telive

To compile:
git clone https://github.com/sq5bpf/telive
cd telive
make
sudo mkdir /tetra
sudo chown YOURUSER.YOURGROUP /tetra
chmod 755 install.sh
./install.sh

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

- tetra codecs

Please read the instructions in osmo-tetra-sq5bpf/etsi_codec-patches , including README_sq5bpf. This shows how to download, patch and compile the codecs. If building by hand put the tetra codecs in /tetra/bin (created in the next step).

To automatically download/compile/install:
git clone https://github.com/sq5bpf/install-tetra-codec
cd install-tetra-codec
Read the README.md file
chmod 755 install.sh
./install.sh

This should download and compile the codec.
It will be installed to /tetra/bin (sudo privileges are needed if /tetra is not writeable by the user).

View the install.sh script to see how it works, change presets etc.

Be sure to read the licensing information for the codecs on the ETSI web page.

tetra-rx reports Air Encryption:1, does this mean that all is encrypted?

No, this means that someone has paid money for the encryption license for their TETRA infrastructure. To use encryption each radio needs to have encryption enabled too, which also costs. So probably there will still be some radios (which are not used for secret communications), without encryption.

How do I find the ppm value quickly?

Although not recommended, this can be done also with this software. Set up everything up as in the "Simple 1 channel setup example", and set the frequency to a known local strong transmitter. This doesn't have to be tetra, I usually use the VOLMET from the local airport for this. Now set the ppm slider so that the channel signal spectrum is symmetrical to 0kHz (in case of AM the carrier should be at 0kHz on the spectrum graph). The ppm value drifts with temperature, so determining the ppm should be done after the receiver has warmed up after at least 10 minutes.

The receiver doesn't work, no matter what frequency I tune it to. Gnuradio complains about PLL unock

You are probably not using the right format. Gnuradio expects to have frequency input in Hz. You can also use prefixes like k for kilo, M for mega etc. Quick example in the 1-channel demo: to listen to 435.2125MHz, you could use 435MHz baseband frequency and 212.5kHz offset (or for example 436MHz baseband and -787.5kHz offset). To tune enter 435M (not 435MHz!) for baseband frequency and press Enter, enter 212.5k for offset.

I get a warning about terminal size in the status window

The terminal size is different than 203x60.The program will still work, but the display may be mangled.

I get "PLAYBACK PROBLEM!! (fix tplay)" in the status window

There is a problem running tplay for live listening. Most probably codecs are not avaliable, there is some permissions problem etc. Fix tplay, and verify that it works by playing the acelp test file, and restart telive.

I get "Too much changes. Are you monitoring only one cell?" in the status window

The network parameters MCC/MNC/LA/Colour Code/frequencies are changing too fast. This is common in a multi-channel setup, where the channels are from different cells. Please use one telive instance to monitor only channels from one cell, otherwise it will get confused. For monitoring multiple cells multiple telive instances can be used (this is described in the "Four channel, two networks setup" exmaple).

I get errors about the device being claimed by another driver

Probably the dvb_usb_rtl28xxu driver is loaded, this is the driver that can be used to watch TV

using these dongles. To disable the module loading create a file /etc/modprobe.d/rtlsdr.conf containing the text:
blacklist dvb-usb-rtl28xxu
After this reboot

I get strange errors, I'm using Ubuntu

Some people report that a reboot solves these problems.
Ubuntu is a strange case generally, because it is preferred by non-technical users, and it is hard to say if these are real problems caused by the distribution itself, or if they stem from the users' lack of ability to use the system and to RTFM.

I installed another gnuradio version and everything broke. How do I fix it?

The easiest way would be to reinstall the system. This can also be caused by installing a package that has gnuradio as a dependency (gqrx etc).

I have trouble when downloading software using the build-gnuradio script

Either the repository is not available at this time, or you have some Internet connectivity problems. Wait a few hours and try again.

On which frequencies can I find tetra signals?

According to wikipedia, in Europe the downlink signals are on:
emergency services:
390-395MHz
commercial services:
395-400MHz
420-430MHz
460-470MHz
915-933MHz

Countries outside of Europe will probably have different bandplans.

OMG! Someone can listen to my secret tetra transmissions! 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, then one can assume that it is a conscious choice of the system designer, and that the system is intended to be open to monitoring.

Of course 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). And since there are not public, they were not subjected to public scrutiny, and may contain weaknesses (as was with GSM A5/1 and other algorithms). Please consider that all your encrypted traffic might be recorded today, and broken a few years from now. This is true of any technology, not just tetra.

But publishing this software makes it possible to monitor something that was not

“monitorable” before. This changes everything

Well, actually not. The protocol specifications were publicly available for a long time and osmocom-tetra code has been available since 2011, patches to record audio have been floating around, and people had developed their own private versions. Also there is commercial software that enables TETRA monitoring like w-code from Wavecom. Publishing this code just makes the process less magical, and security hates magic.

Changelog:

20150824: updated to version 1.5 , added new install info etc --sq5bpf
20150228: added info what TETRA_KML_INTERVAL is for --sq5bpf
20150227: added info about version 1.0 and using location information --sq5bpf
20150130: added info about version 0.9 --sq5bpf
20141208: added info about testfile.acelp --sq5bpf
20141207: added info about telive_1ch_simple.grc --sq5bpf
20141207: added a bit to the FAQ --sq5bpf
20141206: Changed telive version to 0.8, added a bit to the FAQ --sq5bpf
20141127: Added the FAQ. Clarified a few things –sq5bpf