How to stream into Freenet

Dr. Arne Babenhauserheide

<2020-11-30 Mo>

With 1489 Freenet will gain a filter for m3u files — playlists — which enables its use as streaming platform.

In this article I want to show you how you can forward an arbitrary stream into Freenet.

Watching this stream requires a Freenet node running at least build 1489-pre1! For uploading the stream you need pyFreenet. And GNU Linux, but you guessed that, right?

- Install Freenet: `install-freenet-linux.org`
- Install pyFreenet: Get Python3, then: `pip3 install --user pyFreenet3`

The stream has around 10 minute delay, if your node is fast enough. Given that a practical streaming setup for semi-professional free streaming uses around 20s delay, that's not too shabby over a confidential decentralized network. With a stronger internet connection, you might be able to push this further down. I'm limited to 40Mbps in theory, the node for this test is set to 400KiB/s, though, and it does not saturate that.

As source I use theradio.cc, a radio station that streams creativecommons licensed media.

To get gap-less playback, almost without artefacts, use `mpv --prefetch-playlist`.

To stream live, you need a sufficiently fast connection to upload 350KiB **to Freenet** in 30s.

On my node this is not real-time yet: It takes around 11 minutes to insert 3 minutes of music.

The following code uses some optimizations: The uploaded segments fit into a single SSK splitfile (around 300KiB) and multiple files are inserted in parallel to reduce the impact of wait times for detecting successful uploads

(the first with higher priority to reduce the time until the stream can start). It still has room for improvement, though. It could, for example, use multiple insert-nodes, one per file in a batch, and reducing the batch-size to 5. That might allow pushing the delay down to 2 minutes.

```
HOST=127.0.0.1
FREDPORT=8888
FCPPORT=9481
KEY=$(fcpgenkey -H $HOST -P $FCPPORT | tail -n 1)
PUBKEY=$(fcpinvertkey -H $HOST -P $FCPPORT $KEY)
FILEPREFIX="theradiocc-"
rm -rf /tmp/FREESTREAM-split/
mkdir /tmp/FREESTREAM-split/
cd /tmp/FREESTREAM-split/
for i in {000..999}; do
    echo "${FILEPREFIX}${i}.mp3" >> stream.m3u;
done
# insert stream.m3u with compression because it is large and very
# repetitive, and with high priority to ensure that it does not get
# drowned by later mp3 inserts
fcpupload -H $HOST -P $FCPPORT -p 2 ${KEY}stream.m3u stream.m3u
# provide the link to the playlist
echo Streaming radio to $(fcpinvertkey -H $HOST -P $FCPPORT $KEY)stream.m3u
# run mp3 encoding asynchronously for one hour, re-encode to quality 0
# (lowest quality) so this can work with an OK connection.
timeout 60000 nohup ffmpeg -i http://mp3.theradio.cc/  \
-c copy -map 0 \
-segment_time 00:00:20 \
-f segment \
-reset_timestamps 1 "${FILEPREFIX}%03d.mp3" &
# show a command how to actually start the stream
echo you can listen to the stream with
echo mpv --ytdl=no --prefetch-playlist http://$HOST:$FREDPORT/freenet:$(fcpinvertk
for i in {00..99}; do
    # upload 9 files in parallel, because Freenet does some checking
    # for availability, that does not consume much bandwidth but takes
    # time
    # wait until all the next 10 files are available
```

```
    while test ! -e "${FILEPREFIX}${i}9.mp3"; do sleep 10; done
    # wait for one more file to avoid partial files
    sleep 30
    date
    # upload the first of the batch with higher priority
    j=0
    fcpupload -H $HOST -P $FCPPORT -p 2 "${KEY}${FILEPREFIX}${i}${j}.mp3" "${FILEP
    # upload 8 more files
    for j in {1..8}; do
# insert segments in parallel to reduce the impact of wait times
fcpupload -H $HOST -P $FCPPORT "${KEY}${FILEPREFIX}${i}${j}.mp3" "${FILEPREFIX}${i
    done
    # wait for the upload of the last one of the batch as primitive limitation for
    j=9
    fcpupload -w -H $HOST -P $FCPPORT "${KEY}${FILEPREFIX}${i}${j}.mp3" "${FILEPRE
done
```

If you have Freenet 1489-pre1 running at port 8888, you can listen in on a test-stream with mpv:

```
mpv --ytdl=no --prefetch-playlist http://127.0.0.1:8888/freenet:SSK@d8lB5dqTaAt~39
```

(also works in vlc, but with small gaps between the segments)

We have Freenet Radio!

It's imperfect and you'll hit some gotchas, but it works, and that's seriously cool!

Multi-node version

To speed this up more, you can install multiple nodes.

First install a template node:

```
FREENET_NODES_BASEFOLDER=/tmp/freenet-streaming-nodes
mkdir "${FREENET_NODES_BASEFOLDER}"
cd "${FREENET_NODES_BASEFOLDER}"
mkdir node1
cd node1
# following https://www.draketo.de/software/install-freenet-linux.html
```

```
wget 'https://github.com/freenet/fred/releases/download/build01489/new_installer_o
java -jar new_installer_offline_1489.jar -console
```

follow the prompts, then open the Freenet web interface:

```
lynx 127.0.0.1:8888
# follow the wizard
```

Now **stop the node**,

```
cd "${FREENET_NODES_BASEFOLDER}"/node1; ./run.sh stop; cd "${FREENET_NODES_BASEFOL
```

Next just copy the folder:

```
for i in {2..5}; do cp -r node1 node$i; done
```

Then remove the listen-port lines in `freenet.ini`. This causes Freenet to create unique listen ports on the next run.

Also adjust the fcpport, fred port, and optionally the bandwidth:

```
for i in {1..5}; do
    sed -i s/fproxy.port=.*/fproxy.port=8${i}80/  node$i/freenet.ini
    sed -i s/fcp.port=.*/fcp.port=9${i}80/  node$i/freenet.ini
    sed -i s/node.listenPort=.*// node$i/freenet.ini
    sed -i s/node.opennet.listenPort=.*// node$i/freenet.ini
    sed -i s/node.inputBandwidthLimit=.*/node.inputBandwidthLimit=70k/ node$i/free
    sed -i s/node.outputBandwidthLimit=.*/node.outputBandwidthLimit=70k/ node$i/fr
done
```

And start them:

```
for i in {1..5}; do cd node$i; ./run.sh start; cd -; done
```

```
$ grep -i \\.port node*/*ini
node1/freenet.ini:fproxy.port=8188
node1/freenet.ini:fcp.port=9180
node2/freenet.ini:fproxy.port=8288
node2/freenet.ini:fcp.port=9280
node3/freenet.ini:fproxy.port=8388
node3/freenet.ini:fcp.port=9380
node4/freenet.ini:fproxy.port=8488
node4/freenet.ini:fcp.port=9480
node5/freenet.ini:fproxy.port=8588
node5/freenet.ini:fcp.port=9580
```

Now we have 5 running nodes on FCP ports 9180 to 9580. We'll use them to insert:

```
SOURCE=http://mp3.theradio.cc/
KEY=$(fcpgenkey -P 9180 | tail -n 1)
PUBKEY=$(fcpinvertkey -P 9180 $KEY)
FILEPREFIX="theradiocc-"
rm -rf /tmp/FREESTREAM-split/
mkdir /tmp/FREESTREAM-split/
cd /tmp/FREESTREAM-split/
for i in {000..999}; do
    echo "${FILEPREFIX}${i}.mp3" >> stream.m3u;
done
# insert stream.m3u with compression because it is large and very
# repetitive, and with high priority to ensure that it does not get
# drowned by later mp3 inserts
# use the last node for the streaming file
fcpupload -P 9580 -p 2 ${KEY}stream.m3u stream.m3u
# provide the link to the playlist
echo Streaming radio to $(fcpinvertkey -P 9180 $KEY)stream.m3u
# run mp3 encoding asynchronously for one hour, re-encode to quality 0
# (lowest quality) so this can work with an OK connection.
timeout 60000 nohup ffmpeg -i ${SOURCE}  \
-c copy -map 0 \
-segment_time 00:00:20 \
-f segment \
-reset_timestamps 1 "${FILEPREFIX}%03d.mp3" &
# show a command how to actually start the stream
echo you can listen to the stream with
echo mpv --ytdl=no --prefetch-playlist http://127.0.0.1:8888/freenet:$(fcpinvertke
for i in {00..99}; do
    # upload 9 files in parallel, because Freenet does some checking
    # for availability, that does not consume much bandwidth but takes
    # time
    PRE="${FILEPREFIX}${i}"
    # wait until the first 5 files are available
    while test ! -e "${PRE}5.mp3"; do sleep 10; done
    date # for statistics
```

```
    # upload the first 5 files, one file per insertion-node
    # the first 5 get higher priority
    fcpupload -P 9180 -p 2 "${KEY}${PRE}0.mp3" "${PRE}0.mp3"
    fcpupload -P 9280 -p 2 "${KEY}${PRE}1.mp3" "${PRE}1.mp3"
    fcpupload -P 9380 -p 2 "${KEY}${PRE}2.mp3" "${PRE}2.mp3"
    fcpupload -P 9480 -p 2 "${KEY}${PRE}3.mp3" "${PRE}3.mp3"
    fcpupload -P 9580 -p 2 "${KEY}${PRE}4.mp3" "${PRE}4.mp3"
    # wait until the next 5 files are available
    while test ! -e "${PRE}9.mp3"; do sleep 10; done
    # wait some more to avoid partial files
    sleep 60
    fcpupload -P 9180 "${KEY}${PRE}5.mp3" "${PRE}5.mp3"
    fcpupload -P 9280 "${KEY}${PRE}6.mp3" "${PRE}6.mp3"
    fcpupload -P 9380 "${KEY}${PRE}7.mp3" "${PRE}7.mp3"
    fcpupload -P 9480 "${KEY}${PRE}8.mp3" "${PRE}8.mp3"
    # wait for completion of the last upload as a primitive way to limit paralleli
    fcpupload -w -P 9580 "${KEY}${PRE}9.mp3" "${PRE}9.mp3"
done
```

Video-Streaming

Wouldn't it be great to replace youtube with fully decentralized streaming? Can Freenet do it? Are you up for the test? :-)

You won't be live-streaming yet, bandwidth does not suffice with the currently supported codecs, but you can provide video that starts playing right-away.

We'll prepare the nodes just like before, and choose a video file to stream.

```
SOURCE=/tmp/36C3_-_Climate_Modelling-Q_ysS7C8IBw.mkv
KEY=$(fcpgenkey -P 9180 | tail -n 1)
PUBKEY=$(fcpinvertkey -P 9180 $KEY)
FILEPREFIX="36c3-climate-"
rm -rf /tmp/FREESTREAM-split/
mkdir /tmp/FREESTREAM-split/
cd /tmp/FREESTREAM-split/
for i in {000..999}; do
    echo "${FILEPREFIX}${i}.ogv" >> stream.m3u;
```

```
done
# insert stream.m3u with compression because it is large and very
# repetitive, and with high priority to ensure that it does not get
# drowned by later ogv inserts
# use the last node for the streaming file
fcpupload -P 9580 -p 2 ${KEY}stream.m3u stream.m3u
# provide the link to the playlist
echo Streaming radio to $(fcpinvertkey -P 9180 $KEY)stream.m3u
# run ogv encoding asynchronously for one hour, re-encode to quality 0
# (lowest quality) so this can work with an OK connection.
timeout 60000 nohup ffmpeg -i ${SOURCE}  \
-c:v libtheora -c:a libvorbis -map 0 \
-q:v 1 -q:a 1 \
-vf scale=640:480 \
-segment_time 00:00:15 \
-f segment \
-reset_timestamps 1 "${FILEPREFIX}%03d.ogv" &
# show a command how to actually start the stream
echo you can listen to the stream with
echo mpv --ytdl=no --prefetch-playlist http://127.0.0.1:8888/freenet:$(fcpinvertke
for i in {00..99}; do
    # upload 9 files in parallel, because Freenet does some checking
    # for availability, that does not consume much bandwidth but takes
    # time
    PRE="${FILEPREFIX}${i}"
    # wait until the first 5 files are available
    while test ! -e "${PRE}5.ogv"; do sleep 10; done
    date # for statistics
    # upload the first 5 files, one file per insertion-node
    # the first 5 get higher priority
    fcpupload -P 9180 -p 2 "${KEY}${PRE}0.ogv" "${PRE}0.ogv"
    fcpupload -P 9280 -p 2 "${KEY}${PRE}1.ogv" "${PRE}1.ogv"
    fcpupload -P 9380 -p 2 "${KEY}${PRE}2.ogv" "${PRE}2.ogv"
    fcpupload -P 9480 -p 2 "${KEY}${PRE}3.ogv" "${PRE}3.ogv"
    fcpupload -P 9580 -p 2 "${KEY}${PRE}4.ogv" "${PRE}4.ogv"
    # wait until the next 5 files are available
    while test ! -e "${PRE}9.ogv"; do sleep 10; done
    # wait some more to avoid partial files
```

```
    sleep 60
    fcpupload -P 9180 "${KEY}${PRE}5.ogv" "${PRE}5.ogv"
    fcpupload -P 9280 "${KEY}${PRE}6.ogv" "${PRE}6.ogv"
    fcpupload -P 9380 "${KEY}${PRE}7.ogv" "${PRE}7.ogv"
    fcpupload -P 9480 "${KEY}${PRE}8.ogv" "${PRE}8.ogv"
    # wait for completion of the last upload as a primitive way to limit paralleli
    fcpupload -w -P 9580 "${KEY}${PRE}9.ogv" "${PRE}9.ogv"
done
```

This has some glitches at the beginning, but you can already watch it:

```
mpv --ytdl=no --prefetch-playlist http://127.0.0.1:8888/freenet:SSK@I3yBAkjMtFmZPQ
```

And with that we have video-streaming over Freenet!

Split a video file by silence

Depending on your player, playback might not be completely gapless. You can lessen this problem by splitting the video file by silence. While this will likely fail catastrophically for music, it should work pretty well for recorded talks.

Here's a one-line example from Vi at Stackoverflow, adapted to prouce scaled down ogg theora for Freenet, splitting at places with at least 0.3 seconds of -40dB signal:

```
ffmpeg -i infile.mkv -filter_complex "[0:a]silencedetect=n=-40dB:d=0.3[outa]" -map
```