Pasquale Mainolfi

Appunti per il corso di Musica Elettronica

AUDIO OVER NETWORK

...some definition

MTU (Maximum Transmission Unit): “Maximum Transmission Unit” is a measure of the maximum size of data packets that can be transmitted over a network or communication channel without being fragmented. In other words, it represents the maximum size of packets that can be sent through a network without being divided into smaller packets. MTU is measured in bytes.
It is advisable, for the smooth operation with common decoders or to avoid issues such as loss of synchronization or visual artifacts (such as pixelation or dropped frames) during decoding, that the size of the audio or video stream generated by FFmpeg be a multiple of 188 bytes.
This requirement can be important when working with specific data transmission protocols or network devices that handle data in 188-byte packets. For example, the MPEG-TS (Transport Stream) transport format, often used for broadcasting video on television networks, uses 188-byte packets. Maintaining the stream size as a multiple of 188 bytes helps ensure that packets are transmitted without fragmentation and can be decoded correctly.

Bit Rate: the amount of data that is stored in the sound file you are listening to. Every audio file has a “bitrate” associated with it. Every second of an audio recording contains a certain amount of data or bits. The more kilobits per second the greater the quality of the sound. For most general listening 320kbps is ideal.

$\text{Bit Rate (Mbps)} = \text{sample rate} \cdot \text{bit depth} \cdot 2$

$\text{File Size (MB)} = \frac{\text{bit rate * seconds}}{8}$

References

- UDP User Datagram Protocol: https://en.wikipedia.org/wiki/User_Datagram_Protocol
- RTP Real-time Transport Protocol: https://en.wikipedia.org/wiki/Real-time_Transport_Protocol

- FFMPEG Documentation: https://ffmpeg.org/ffmpeg.html#Generic-options

- FFMPEG Real-Time Audio/Video Streaming: https://trac.ffmpeg.org/wiki/StreamingGuide

- Capturing Desktop: https://trac.ffmpeg.org/wiki/Capture/Desktop
- Jack Audio Connection Kit: https://jackaudio.org/
- Jacktrip: https://jacktrip.github.io/jacktrip/
- Jacktrip Manual: https://www.mankier.com/1/jacktrip
- QjackCtl: https://qjackctl.sourceforge.io/qjackctl-downloads.html#Dow nloads

FFMPEG

MAIN AUDIO OPTIONS

Generic options

-formats show available formats
-devices show available devices
-codecs show all codecs
-encoders show all encoders
-decoders show all decoders
-protocols show all protocols
-filters show all filters

-i path to input file

Main audio options

-ar set audio rate
-ac set the number of the audio channels
-c:a set audio codec (format, see ffmpeg -formats)
-b:a set bit rate
-ss specify start time (hh:mm:ss.000)
-t duration of segment to extract (hh:mm:ss.000)
copy copy the original value

Example

1. Convert .mp3 to .wav format with sample rate equal to 44100 Hz

```
ffmpeg -i <path_to_audio_file.mp3> -ar 44100 <path_to_audio_file_out.wav>
```

2. Convert .wav file with pcm signed 16 bit little endian, with four channels and sample rate equal to 44100 Hz

```
ffmpeg -i <path_to_audio_file.wav> -c:a pcm_s16le -ac 4 -ar 44100 \
          <path_to_audio_file_out.wav>
```

3. Extract a segment with a certain duration from an audio file

```
ffmpeg -i <path_to_audio_file.wav> -ss 00:00:30.000 -t 00:02:00.000 \
          -ar copy -c:a copy <path_to_audio_file_out.wav>
```

Streaming options

Audiotoolbox
Print the list of supported devices and output a sine wave to the default device

```
ffmpeg -f lavfi -i sine=r=44100 -f audiotoolbox -list_devices true -
```

(-) means that the output field can be empty to refer to the default system output or a number that refers to the device index as shown using: **-list_devices true**. Alternatively, the audio input device can be chosen by index using the **-audio_device_index** , overriding any device name or index given in the input filename. All available devices can be enumerated by using **-list_devices true**.

or

```
ffmpeg -list_devices true -f avfoundation -i dummy
```

avfoundation is for MacOs!

ALSA
Play on default ALSA

```
ffmpeg -i INPUT -f alsa default
```

Play a file on soundcard 1, audio device 7

```
ffmpeg -i INPUT -f alsa hw:1,7
```

From Source to Destination
RTP Source:

```
ffmpeg -f avfoundation -i hw:0 -ac 1 -c:a pcm_s16le \
          -ar -f rtp rtp://<IP>:<PORT>
```

RTP Destination:

```
ffplay rtp://<IP>:<PORT>
```

UDP Real-Time Source:

```
ffmpeg -re -i <INPUT> <PARAMETERS> -f wav udp://<IP>:<PORT>
```

UDP Real-Time Destination:

```
ffplay -i udp://<IP>:<PORT>
```

Microphone stream: WAV format:

```
ffmpeg -re -f avfoundation -i ":0" -ar 44100 -ac 1 -c:a \
       pcm_s16le -f wav udp://localhost:12345
```

or

```
ffmpeg -f avfoundation -i ":0" -ar 44100 -ac 1 -c:a pcm_s16le \
       -f wav "udp://localhost:12345?pkt_size=1316"
```

...to reduce latency:

- set frame size: **-blocksize**
- set real-time buffer size: **-rtbufsize**
- set

Compressed format:

```
ffmpeg -re -f avfoundation -i ":0" -c:a libvorbis \
       -f ogg udp://localhost:12345
```

```
ffmpeg -re -f avfoundation -i ":0" -c:a ac3 -b:a 192k \
       -f wav udp://localhost:12345
```

follow the command that list all available devices (audio and video). -i ":0" means use audio devices at index 0 -re means real-time

```
ffmpeg -f avfoundation -list_devices true -i ""
```

Please note: *avfoundation* is for MacOS, ALSA for Linux

Note thar if you want to use the output as an input (speaker to input), please install any routing app like Soundflower or Loopback.

Reduce latency flags:
-probesize
-fflags
-flags
-analyzeduration

ffmpeg -> Python:

```
ffmpeg -re -i <INPUT> -c:a pcm_s16le -ar 44100 -ac 1 \
       -f wav "udp://<IP>:<PORT>?pkt_size=<SIZE>"
```

in python, using socket and pyaudio:

```
import socket
import pyaudio
import numpy as np

IP = "127.0.0.1"
PORT = 9006

pa = pyaudio.PyAudio()
stream = pa.open(
    format=pyaudio.paInt16,
    rate=44100,
    output=True,
    channels=1
)

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP, PORT))

try:
    while True:
        data, address = sock.recvfrom(4096)
        stream.write(data)
        data_vec = np.frombuffer(data, dtype=np.int16)
        print(f"{data_vec} \n")
except KeyboardInterrupt:
    print("[DONE] Processo terminato!\n")
    exit(0)
finally:
    stream.stop_stream()
    stream.close()
    pa.terminate()
    sock.close()
```

how to know your IP address...

```
ipconfig getifaddr en0
```

Python -> ffmpeg:

```
import wave
import socket
import time


IP = "127.0.0.1"
PORT = 8001

SR = 44100
CHUNK = 4096
INTERVAL = CHUNK / SR

socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

wave_file = wave.open("vox.wav", "rb")
sample_rate = wave_file.getframerate()


while True:
    time.sleep(INTERVAL)
    frame = wave_file.readframes(CHUNK)
    socket.sendto(frame, (IP, PORT))

    if not frame:
        break


socket.close()
wave_file.close()
```

```
ffplay -f s16le -ar 44100 -ac 1 udp://<IP>:<PORT>
```

Csound -> ffmpeg:

```
<CsoundSynthesizer>
<CsOptions>
-o dac -b 2048 -B 4096
</CsOptions>
<CsInstruments>

sr = 44100
ksmps = 1
nchnls = 2
nchnls_i = 1
0dbfs = 1

#define IP #"127.0.0.1"#
#define PORT #8100#


instr 1

a1 = poscil(.5, 220)
socksend(a1, $IP, $PORT, 10)

endin



</CsInstruments>
<CsScore>

i 1 0 60


</CsScore>
</CsoundSynthesizer>
```

```
ffplay -f f64le "udp://127.0.0.1:8100"
```

-f f64le: Csound send 64bit float point

JACKTRIP

MAIN FLAGS

JACK AUDIO **-d** master-backend-name
-dhw, –device hw:device_name specifies the name of the hardware audio device to use when using the `alsa` driver. Replace `device_name` with the name of your hardware device
-p, –period nframes sets the period size (buffer size) in frames. A smaller period size can reduce latency but may require more computational power
-n, –nperiods nperiods sets the number of audio periods. T his affects latency and performance
-r, –rate samplerate sets the sample rate in Hz. For example, `44100` for 44.1 kHz
-s, –server-name server_name specifies the name of the JACK server. Useful when running multiple instances of `jackd`
-X, –no-mlock disables memory locking. Useful if you don't have the necessary privileges to lock memory
-P, –port-max nports sets the maximum number of audio ports
-C, –capture ports specifies a list of ports to be captured by default when JACK is launched
-P, –playback ports specifies a list of ports to be played back by default when JACK is launched

JACKTRIP

-s server in P2P mode
-S server in HUB mode
-c client in P2P mode
-C client in HUB mode
-n number of channels
–**receivechannels** number of receive channels
–**sendchannels** number of send channels
-o port offset
-z set buffer to zero when the underrun occurs
-J, –clientname set client name
-p server hub mode (0, 1, 2, 3, 4, 5)
-T set sample rate
-F set buffer size
-R, –realtime (without Jack) ese system's default sound system instead of Jack
–**audiooutputdevice** (without Jack) set audio device to use; if not set, the default device will be used –**listdevices** list available audio devices

STREAM

Terminal 1: start jack

```
jackd -d <DEVICE>
```

Terminal 2 (test): open Jacktrip server (HUB MODE)

```
jacktrip -S -p<HUB PATCH MODE> -n<NUMBER OF CHANNEL>
```

Terminal 3 (test): open Jacktrip client (HUB MODE)

```
jacktrip -C <IP> -n<NUMBER OF CHANNEL> -T <SAMPLE RATE> -F <BUFFER SIZE>
```

for P2P mode, use -s and -c.

now, …open QjackCtl and connect!

Note: Sample rate and buffer size must the same (client and server).