Openvibe data Broadcasting - Documentation

Seidi Yonamine*

July 2019

Abstract

This document introduces and shows requirements and use case of a server-client system to broadcast Openvibe acquired signal through TCP/IP protocol, which can be accessed from anywhere in the world online and in real-time. Files repository is on openvibe-broadcast

1 Requirements

The system works with two ends, and the following should be installed on them (some observations follow, for implementation methodology):

Server:

- OpenVibe (v2.0 and above, which has a TCP Writer box): implements the acquisition and transmition of brain signals. A test scenario is provided in the repository
- UPnP PortMapper

Obs.: Portmapper is a solution that can implement port forwarding, basically exposing your IP address to the internet. you can try and implement safer solutions to make your server visible to the client side

Client:

- Python 3

Obs.: Some package installations are required, so I recommend installing Anaconda

*Any feedback is appreciated and can be sent directly to my email: seidi.yamauti@gmail.com

2 Test Project Components

A repository was made with a test project. from the server side, an OpenVibe scenario produces sinusoidal waves and transmitts them via TCP communication to a specified port and UPnP PortMapper will forward the port so it can be visible through the internet

The client side should run a python code, which implements socket to receive streaming data given TCP protocol. All communication parameters were left to be filled in a configFile.txt

2.1 [SERVER] signal_broadcast.msx

If you open the signal_broadcast.msx file on OpenVibe v2+, it will generate a scenario containing boxes *Sinus oscillator*, which produces sinusoidal waves (parameters: Channel count, Sampling frequency, Generated Epoch Sample Count), *TCP Writer*, which streams data through a port and PC IP, and *Signal display*, to show the generated waves.

The scenario is shown on figure 1

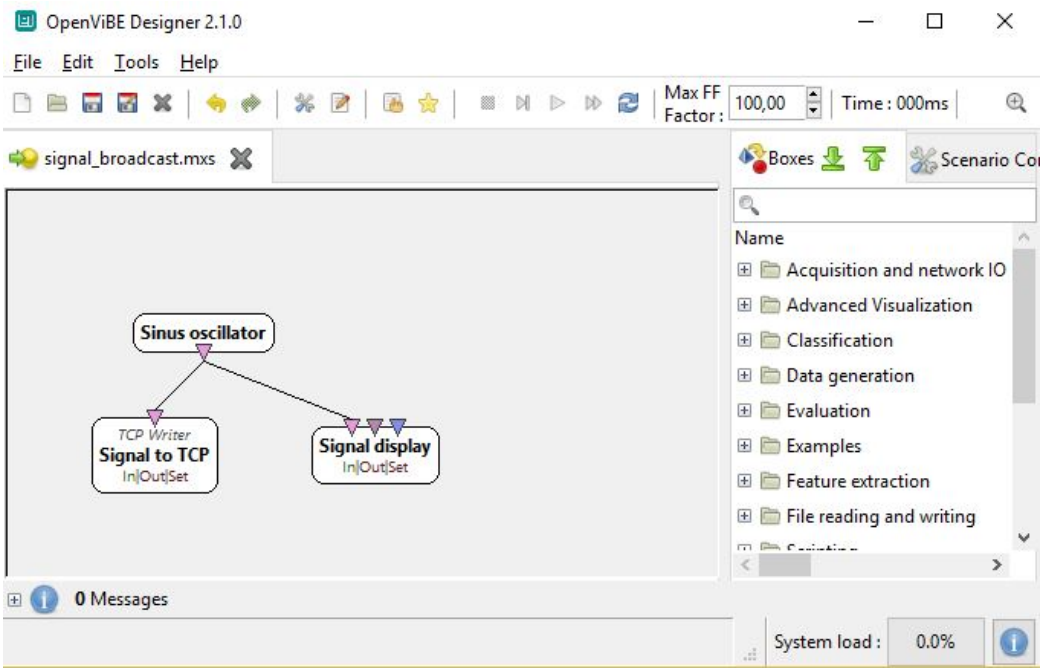

Figure 1: Contents of signal_broadcast.msx as opened in OpenVibe 2.1.0

To initialize it, double-click on box Sinus oscillator and set Channel Counts (e.g.: 16), Sampling Frequency (e.g.: 256) and Generated Epoch

Sample (e.g.: 16) and press Apply. Note those values, as they should be filled in the configFile.txt Also, set a port on TCP Writer box (default is 5678)

2.2 [SERVER] UPnP Portmapper

Portmapping is a solution I used to forward the TCP/IP port such that it is visible beyond the local net, and then a client with access to the internet can reach it. A solution using VPN is recommended for safety issues

A software that does portmapping is used, it is illustrated on 2

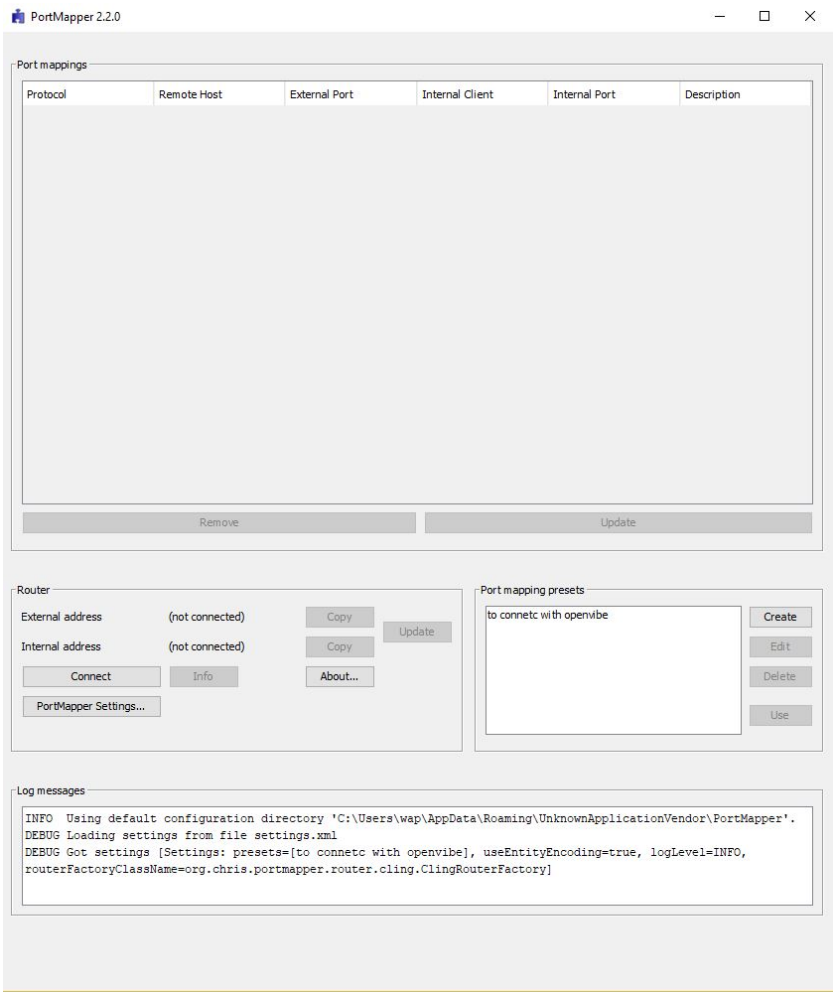

Figure 2: PortMapper interface

Click on Connect and it should route your local IP to the external IP. Also, I creted a preset port mapping called *to connect with openvibe*, External

Port = 5006 and Internal Port = 5678 (the logic is that every message coming to port 5006 of public IP will be forwarded to port 5678 of internal IP)

2.3 [CLIENT] test_broadcast.py and configFile.txt

The code imports a list of python packages, the main ones being socket and matplotlib. Socket will create a client-side socket object that access the server IP and Port and continually receives streaming data (that is implemented on a thread), the code then process it and plot real-time raw data and power-spectrum in 2 separate windows. This code shouyld be run from prompt, using *python test_broadcast.py* command.

The configFile.txt then serves as a list of parameters for communication and visualization which will be input to the code. An example of this file is shown on figure 3

```
################################
#     Streaming Parameters     #
################################

# Server's IP and socket
SERVER_IP = 127.0.0.1
SOCKET    = 5555

# Values come from OpenVibe
CHANNELS         = 8
ACQUISITION_FREQ = 512
EPOCHS           = 16

################################
#   Visualization Parameters   #
################################

# Change it as you wish to get a better visualization

WINDOW = 4 # seconds
DO_FFT = 1 # 1 = YES, 0 = NO
```

Figure 3: Information on configuration file

Set SERVER_IP to the public IP from the server computer, and SOCKET as the Port number set on PortMapper (5006 in the example).

Values CHANNELS, ACQUISITION_FREQ and EPOCHS are the values set on OpenVibe (Channel count, Sampling frequency, Generated Epoch Sample Count, respectively).

Finally, WINDOW is the range of seconds to show raw data in the plots and DO_FFT is a control flag to process (1) or not (0) fft on the data and show it

Figure 3 shows the two windows that plot the above mentioned real-time data

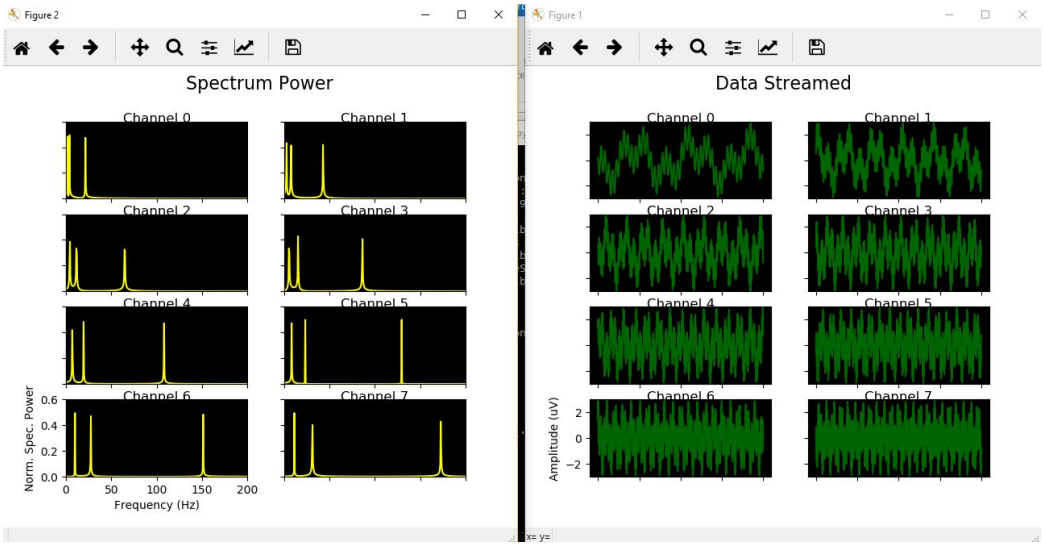

Figure 4: Power Spectrum and Raw Data being plotted in real-time