Networked Music performance: A step-by-step guide to real-time musical collaboration over the internet

Compiled by @benloveridge - Last updated March 2020

Introduction

Networked music performance (NMP) enables remote musicians to interact and perform in real-time over the internet through the correct configuration of network, computer, audio and software systems.

Who is this document for

This document is a 'how-to' guide to achieving NMP as well as a deeper explanation for performers, educators and technicians wishing to achieve low-latency musical collaboration over the internet.

Definitions

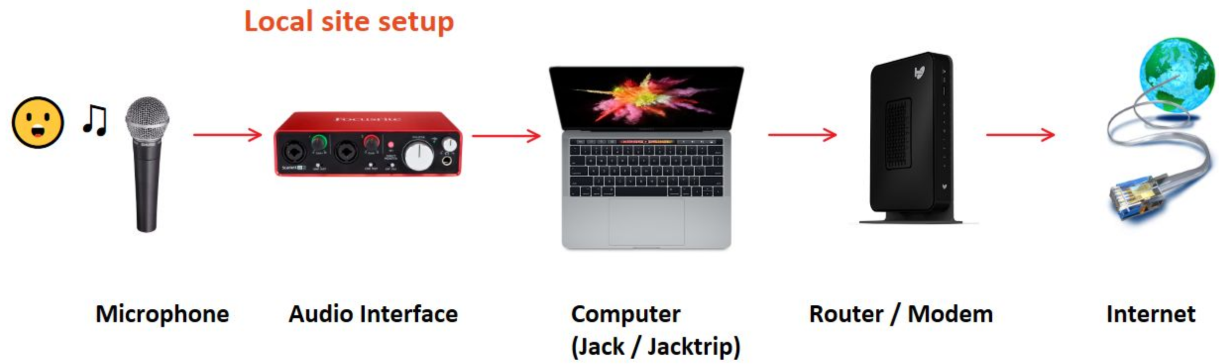
- By *real-time*, we are referring to a latency of 25 milliseconds or less between two geographically separated locations. This time difference between the creation of the sound at one site, to when it is heard at another other location is around the time limit that musicians can play together at a moderately fast tempo.
- By *musical collaboration* we are referring to when both sides can hear each other as if they are in the same physical room, and can respond instantly to musical cues such as a melodic change or a fluctuating tempo

Acknowledgments

This research was undertaken as part of the Melbourne Networked Society Institute seed funded project 'Music Therapy in Virtual Environments' at the University of Melbourne. Much of the information in this guide was created from notes as part of the Kadenze course Online Jamming and Concert Technology as well as on-site testing on-site between July 2015 and July 2017.

Equipment Requirements

The audio from each site is sent through the hardware setup via the internet to the remote site and monitored via headphones at each end.



- Microphone - Dynamic or condenser
- XLR cable - to connect microphone to audio interface
- Audio Interface - needs to have a direct monitoring option and ideally low latency features (in our testing we're using the 2nd Generation Focusrite Scarlett 2i2 via USB). The Rodecaster Pro has also been tested to work down to Frames/Period values of 16 and has the added bonus of having internal multitrack recording capabilities.
- Headphones - for monitoring the direct and remote signal through the audio interface
- Computer - We've found Mac OS X to be the ideal platform (PC still being tested)
- Ethernet cable - connects the computer to the modem/router. Make sure you turn off Wifi on your computer as the audio will need to travel over ethernet.
- Modem/Router - sends the signal from the computer to the outside world
- Network Cable - to connect computer to the modem/router
- Internet - NBN or cable connection with an upload bandwidth of at least 3Mbps. Check using Speedtest: http://www.speedtest.net

Network Setup

Education institutions usually have inbound firewall restrictions that prevent Jacktrip traffic from getting through to computers on the internal network. To run a Jacktrip server, it is necessary to get an ethernet port configured to allow for a static IP address (usually possible on a research network). You will need to liaise with your network operations team for configuration assistance.

To run a Jacktrip server from a home site you will need to do the following steps:

- Connect your computer to the modem/router via an ethernet cable
- In your computer network settings, make a note of your local IP address
- Go into your router settings and find the port forwarding area
- Set the port forwarding to your local IP address as UDP port 4464

- Make a note of your public IP address (this information will be used to send to the remote site)

Running Jacktrip in client mode should not require any special port configuration from either a home or institutional location. (How can this be?)

Software Setup - Getting Started:

- Install Jack Audio
- Install JackTrip (Terminal version)

Other installs

- (Optional) Install Jacktrip (GUI version Yunhan Yi - Beta)
- Install Jacktrip from .pkg (Mac)
- Install Jacktrip (Windows)

Documentation

- See CCRMA jnstall software guide
- There is also a Jacktrip Manual

From the newly installed Jack folder in Applications, drag the **qjackctl** application into the dock. Do the same for the terminal window, located in the Utilities folder in Applications.

Optional (for iPerf testing only)

- Install Audacity
- For iPerf installation - Follow the instructions to install **Homebrew** at http://brew.sh
- After installation is complete, in the terminal window type *brew install iperf*

Testing your audio settings

- Plug in your USB audio device to the computer and connect a microphone to the device
- Open the **qjackctl** application and click *setup*
- Set the server path to */usr/local/bin/jackd*
- Select *coreaudio* in the Driver setting
- Select the USB device in *interface*, set sample rate to 48000 and frames/period to 16, click *ok*
- Click *Start* (If it crashes the first time, try again)
- Select *Connect,* then click on *system* on both sides of audio menu and click *connect*

- Confirm that Direct Monitor is turned off or down on your audio interface and talk into the microphone. You should be able to hear your audio in the left hand side ear of your headphones
- To hear audio from both sides of your headphones, expand each system menu using the drop down triangle, click on *capture 2* and *playback 2,* then click disconnect, click on capture_1 and click Connect. This will connect the left channel microphone input to the right channel playback.
- Select *system* on both sides then click *Disconnect All*

Recording Audio (optional)

- Open *Audacity*
- In the drop-down menu next to the microphone symbol, select *Jack Router*
- In the drop-down menu next to the speaker symbol, select your audio device
- Press record, speak into the microphone and confirm input levels are appropriate
- Repeat for any second instrument
- Note: You will not hear your live audio back through your headphones
- To enable this, connect system to system and to enable input sources in each ear, connect capture_1 to playback_2 and capture_2 to playback_1
- If you look at the Connections window in the Jack application, two lines connecting the system audio will be linked to the two inputs of the Audacity application
- Stop the recording. You will notice the Audacity application disappears from the Jack Connections window. This is the nature of how Audacity interfaces with Jack.
- Playback the recorded audio to confirm that sound can be heard through your headphones

Network Setup - Preparing for two-way low latency connection

- Turn off any wireless connections and check your wired connection to the router is active
- Optional: Check network configuration in settings or by typing in the terminal command *ifconfig -a* (OSX)
- Do a local ping self test - In the terminal type *ping* 127.0.0.1
- The time should be around 0.05 ms on average. (About the equivalent of the time of two samples at 48kHz)
- Ping the gateway on your local network - *ping* [*local gateway IP].* On a wired connection the time should be around 0.2ms. Note: Using an Ethernet over Power adapter will return a time of between 3-5 ms and is not recommended. Using WiFi is also not recommended.
- Ping another computer on the same local network *ping* [*local computer IP].* The returned time should be around 0.3 ms
- Ping an external IP address eg - *ping google.com.* If the ping fails then ICMP could be blocked (not all sites allow being pinged), try a different site if this occurs

Setting up the router for Jacktrip

- Check your local IP address via the network setting of *system preferences*
- On your local router, select port forwarding and open port 4464 for the local IP address you are using. If using more than one local computer for testing, you can use port triggering to open the port for all computers on your network.
- You can run your local computer as an iperf or jacktrip server by checking the IP address at https://whatismyipaddress.com/. Keep in mind that if you do not have a static IP, your IP address may change from time to time.

Notes - Even without setting up the router, it has been noted that some connections have been successful. This needs further investigation.

Network testing with iPerf

Real-time audio is transmitted via UDP and the success of the connection between the two computers can be tested in advance using iPerf. Start by testing iPerf on the same computer, opening the first terminal window as the server and the second as the client.

- Open a terminal window
- Type iperf -u -s -p4464
- Open a new terminal window
- Type iperf -u -c [IP address of server] -p4464 -d -b1M -l512

In the case of a self-test, the IP address of the server will be the same as the computer that is running as the client.

Notes: -u (UDP), -c (Client), IP address, -p (port), -d (duplex) -b (bandwidth in Mbps) -l (byte packet size)

A successful connection result should look something like this when doing a test run from the client computer

```
------------------------------------------------------------
Server listening on UDP port 4464
Receiving 512 byte datagrams
UDP buffer size:  192 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 128.250.xxx.xxx, UDP port 4464
Sending 512 byte datagrams
UDP buffer size: 9.00 KByte (default)
------------------------------------------------------------
```

[5] local 192.168.0.23 port 64249 connected with 128.250.xxx.xxx port 4464
[4] local 192.168.0.23 port 4464 connected with 128.250.xxx.xxx port 56193
[ID] Interval   Transfer   Bandwidth
[5] 0.0-10.0 sec 1.19 MBytes   999 Kbits/sec
[5] Sent 2441 datagrams
[5] Server Report:
[5] 0.0- 9.7 sec 1.18 MBytes 1.02 Mbits/sec   3.156 ms   21/ 2440 (0.86%) sender (from the client to the server)
[5] 0.0- 9.7 sec 1 datagrams received out-of-order
[4] 0.0-10.6 sec 1.19 MBytes   943 Kbits/sec   0.188 ms   0/ 2443 (0%) receiver (from the server to the client)

Port 4464 is the incoming port on either computer

The Results indicate Bandwidth, Network jitter (deviation in time for periodic arrival of datagrams), Packets lost/total

The total number (2443) is how many packets were transferred during the 10 second period.

Note how much worse the jitter is going from the client (at 1Mbps upload link at home) to the server (Uni connection) than the reverse trip

To test the bandwidth limit between the two computers, increase the bit rate of transfer until the percentage loss reduced to around 0.2%. Eg add the term -b[bandwidth] at the end of the command eg -b10M, -b50M, -b100M, -b800M

Using the -l command allows you to specify a lower than default byte packet size (1470) which more closely matches the Jacktrip traffic you would generate and give you more accurate testing results for your bandwidth. By reducing the packet size, you are increasing the rate at which those packets must be sent so expect to see a greater number of errors as you approach the limits of your bandwidth. Knowing what byte packet size causes the error rate to increase above 1% allows you to calculate the ideal frames per packet setting to use in Jacktrip even before you've tested the connection with audio.

Once you successfully run an iPerf test from the same computer, run the server from a different computer on the same network then from a different computer on an external network

Once the network connections and audio devices are all working correctly, you are ready to test Jacktrip.

Running Jacktrip (Mac)

- On the host computer, start qjackctl

- On the client computer, start start qjackctl
- Make sure both machines have the same Frames/Period and Sample Rate. In terminal type: **jacktrip -s**
- In terminal type: **jacktrip -c [server IP address]**
- Both machines should now be connected and passing audio between each one
- For a single channel connections use **jacktrip -s -n1**
- For additional incoming connections, on the server open a new terminal window and add **-o10, -o20 etc** to **jacktrip -s.** On each client machines connect to the server with the additional port info appended.

Troubleshooting

- If audio crackles (these are called xruns - buffer underrun or buffer overrun) try connecting at a higher Frames/Period setting on both computers
- Or use a queue buffer command such as -q(number) on either end
- Input q length is measured in packets (which is the same the buffer)
- Audio Sources - Make sure the audio interface is plugged in when starting Jack
- Packet jitter - use the playback buffer command ie -q100 and reduce until crackles re-appear. By default it is 4 packets in length. Each UDP packet has redundant mini packets which overlap the last sent in case of lost UDP packets
- One buffer / packet can have 128 frames and if dropped cause a gap of 2.66ms in the sound
- Use -z command (Zero underruns) to fill missing packets with mute sound
- Traceroute [IP address] - do in both directions
- Jacktrip server crashes when connected to from clients running on a VPN
- Going through a VPN can add 3-4ms.
- Going through an Ethernet over Power adapter causes extra latency and unreliability
- Make sure you terminate any iperf servers running in terminal before attempting a jacktrip connection otherwise you won't get a connection due to the port 4464 being allocated
- If testing different computers from home make sure you have port forwarded to the correct computer each time otherwise you'll find incoming traffic blocked

Recording Audio from networked computers (OSX)

- With both computers still connected to each other through Jacktrip, open Audacity on the server computer, confirm that 'JackRouter' is still selected as the input and press record
- In the qjackctl click on 'Connect', highlight JackTrip receive_1, Audacity In2 and press Connect. This will router the remote audio into Audacity onto Channel 2.
- If doing a single channel mic recording, select system capture_2 and click Disconnect to close the connection to Audacity in2
- You will now be recording local audio to Audacity channel 1 and remote Jacktrip audio into channel 2 (4-5ms test)

Terminology and explanations

With mono audio, each frame has 1 sample of 16 bit audio (expressed in binary as one of 65,536 levels of energy). Each frame is 2 Bytes.

At 16 frames per period and 3 periods per buffer, 48 samples per buffer at a 48Khz sampling rate gives you a buffer of 1ms. The time separation between samples is 20.83 microseconds. Since each sample is 16 bits, each buffer is 16 * 48 = 768 bits which is being generated each ms. This is equal to a data rate of 768 Kbps.

Once a buffer comes available, it is sent as a UDP datagram carried in a single IP packet. (Note: A UDP packet is commonly referred to as a datagram. Packet is used when referring to TCP)

However, each packet being sent contains both IPv4 (size, source address, destination address etc) and UDP header (Buffer size, sample rate, bit depth, channels, sequence number, timestamp) information. The size of an IPv4 header is 20 bytes and the size of a UDP header is 8 bytes so each packet would add 28 * 8 = 224 bits. The total data size each ms would now be (768 + 224) 992 bits which is equal to 124 Bytes per packet. The total data rate would now be 992 kbps with packets being sent every millisecond.

Testing results - Packets out 3126 data sent/sec 373 KB (2912 kbps) - stereo
Testing results - Packets out 3032 data sent/sec 271 KB (2912 kbps) - stereo
Testing results - Packets out 3762 data sent/sec 439 KB (3512 kbps) - stereo

At 64 frames per period and 3 periods per buffer, 192 samples per buffer at a 48Khz sampling rate gives you a buffer of 4ms. Since each sample is 16 bits, each buffer is 16 * 192 = 3072 bits which is being generated each 4ms. This is still equal to a data rate of 768 Kbps.

Each packet now contains 224 bits of header data plus 3072 bits of data for a total of 3296 bits of data which is being sent every 4 milliseconds (equal to 412 Bytes per packet). This equals a rate of 824 Kbps.

At 128 frames per period and 3 periods per buffer, 384 samples per buffer at a 48Khz sampling rate gives you a buffer of 8ms. Since each sample is 16 bits, each buffer is 16 * 384 = 6144 bits which is being generated each 8ms. This is still equal to a data rate of 768 Kbps.

Each packet now contains 224 bits of header data plus 6144 bits of data for a total of 6368 bits of data which is being sent every 8 milliseconds (equal to 796 Bytes per packet). This equals a rate of 796 Kbps.

Problems occur with jitter if the network can't deliver each packet every millisecond then packets are dropped and the audio sounds crackly. On a LAN network we've tested Mac hardware being able to receive audio at a 16 frames per period rate (fpp). Over the internet on a throttled upload connection we've found at 16 fpp the network jitter is too long to be able to deliver packets constantly every millisecond. However, on a high upload bandwidth connection, audio can be delivered at 16fpp with a a 1ms buffer with clarifty.

By adjusting the settings so that the buffer is slightly larger than the network jitter as tested in iPerf, stable audio can be received.

What is in a packet?

- A *frame* is 1 sample of mono audio data. If the audio is stereo, 1 frame is 2 samples (left + right)
- A *packet* is a group of frames, usually meant to be the set of frames for all channels at a given point in time
- A *buffer* is a group of frames delivered for processing and sent as packets
- Frames per period = Buffer
- A packet is the same as a Buffer

Extra Testing

In terminal typing *Traceroute [IP address]* allows you to see the paths that a network packet travels on to the destination. This may be different either way.

For example: 192.168.0.1 (192.168.0.1) 0.612 ms 0.337 ms 0.217 ms
The results show the three round-trip times to show the variation in time sending.

Resources

- JMess (for Multi-site Configurations) - https://github.com/jcacerec/jmess-jack/releases
- Jacktrip Manual - https://sites.google.com/site/jacktripdocumentation/
- Kadenze Online Jamming course - https://www.kadenze.com/courses/online-jamming-and-concert-technology-vi/info
- PCM terminology
- https://larsimmisch.github.io/pyalsaaudio/terminology.html
- Jacktrip under the hood
- http://quod.lib.umich.edu/cache//b/b/p/bbp2372.2009.115/bbp2372.2009.115.pdf
- Jacktrip list of commands - http://manpages.ubuntu.com/manpages/wily/man1/jacktrip.1.html
- The User Datagram Protocol (UDP) - http://www.erg.abdn.ac.uk/users/gorry/course/inet-pages/udp.html
- Jacktrip Google Groups Forum - https://groups.google.com/forum/#!forum/jacktrip-users

More terminology

- The MTU is the maximum size of an IP packet that can be transmitted without fragmentation. (Max MTU is 1500 Byes or 12,000 Kb or 12Mb).
- Ethernet has an MTU of 1500 bytes.
- An IP packet is composed of two parts: the packet header and the payload.
- The size of an IPv4 header is *at least* 20 bytes, the size of an IPv6 header *at least* 40 bytes.
- The payload of an IP packet is typically a TCP segment or a UDP datagram.
- A UDP datagram consists of a UDP header and the transported data.
- The size of a UDP header is 8 bytes. This means an IP packet with an empty UDP datagram as payload takes at least 28 (IPv4) or 48 (IPv6) bytes, but may take more bytes.

General Info

- Light travels around 200km/ms through fibre

Pulse-code modulation (PCM) is a method used to digitally represent sampled analog signals.

Further discussion and sections to be added

- Reasons for using open-source software compared to third-party providers
- Challenges with integrating real-time audio into the Unity game engine for spatialisation purposes (latency issues)
- Issue with using Windows and low-latency
- Binaural workflow (yet untested but should be possible)
- Omni-binaural (could be possible?)
- Ambisonic workflow (could be possible?) 8 channel with https://hear360.io/shop/8ball
- Method for more than two connections
- More photos and diagrams
- Other real-time audio solutions: JamKazam, Soundjack, SofaSessions
- Can you use Delta encoding for low latency audio transmission?
- Can you send MIDI via NMP? Investigate Open Sound Control (OSC) http://opensoundcontrol.org/introduction-osc as well as chuck and supercollider
- The uncertainty principle as it relates to sound. Visual description of samples in detail and how you can't determine the frequency of a sound from one sample.

Quickstart checklist for testing

- Equipment
- Software

- Speeds
- Download
- Upload
- Port Forward / openness 4464
- iPerf results – each way
- Jacktrip 16, 32, 64, 128
- Quality

Questions:

In loopback mode, do you take into account Jacktrip latency at the far end for timing or does the signal not go through any extra latency?
Not sure. Maybe not?

-l works as loopback mode but -l is listed as byte packet size so which is it?

Q: So why is measured packet arrival twice what is the calculated arrival should be. Eg at 128 frames/period and 2 periods / buffer at 48kHz, the ‘latency’ or ‘buffer’ or packet frequency is 5.33ms. In one second you should expect to see 187 packets (=1/0.00533), however the total received is around 378. Where are all the other packets coming from? This is when channel is n=1. Reducing to n=1 only halves bandwidth, not the number of packets being sent.

A: TBC

Q: What does this refer to in iPerf and why are they different? Server: UDP buffer size: 192 KByte (default) and Client: UDP buffer size: 9.00 KByte (default) (This is equal to 1536 and 72bits of UDP buffer but in an iperf test, how often are packets sent?)

A: TBC

Q: What is the difference between a 1470 byte datagram and the UDP buffer size? Why is server using larger UDP buffer sizes than the client?

```
Server listening on UDP port 4464
Receiving 1470 byte datagrams
UDP buffer size:  192 KByte (default)

Client connecting to ………….., UDP port 4464
Sending 1470 byte datagrams
UDP buffer size: 9.00 KByte (default)
```

A: TBC

Q: Why are Periods/Buffer not able to be changed on Mac and why it's different on different computers)

A: TBC

Q: Is it best to set -q equally on both sides of the connection? I think the answer is yes but I've noticed that you can still connect both sides with different q values. My understanding is the -q setting adjust the buffer of the audio being sent from the computer that sets that q value - so the affected result is actually heard by the other side. Is that right?

A: It can be asymmetrical and it's actually the other way around -- it sets the input queue (aka playback) in number of buffers. I often use a different setting on each side when one path direction is suffering worse.

Q: Also is the -q setting just an arbitrary number of is meant to represent a time interval in ms? I've noticed the affect on latency is around half of the q value if both sides use that figure.

A: It is in number of buffers, so time interval depends on buffer size and sample rate.

Q: If no audio is being created at either end is any bandwidth being used up and sent to either side? I don't see any way of actually measuring the used bandwidth of a jacktrip connection.

A: I measure that graphically using the System Monitor application (shows flows in and out of the host)

Q: In the qjackctl window there is a RT and percentage value that changes over the connection. What do the numbers here mean? I think it's to do with the connection quality? I couldn't find any documentation that describes what the information in the interface actually means.

A: http://lists.linuxaudio.org/pipermail/linux-audio-user/2014-September/099004.html

Q: If you are getting network jitter, wouldn't it be better to adjust your frames per period settings to increase the buffer rather than using the -q (input buffer "cushion")? How are these two things any different?

A: TBC

Q: I thought packet loss was caused by network jitter? If not then what is the difference and what causes it?

A: Packet loss can be helped with the Jacktrip packet redundancy (at the cost of extra bandwidth and latency) -r (extra packets to duplicate possible missing packets)

Q: Audacity recording comes out choppy

A: Alternative is to use a digital recorder out of the line-output of the audio interface box

Notes from Rehearsal Sessions

- Test Jacktrip with Rode Podcaster device - Rehearsals could be done in Podcast mode
- What does network dropout sound like compared to cpu / buffer underun issues
- Test multitrack recording with Jacktrip
- Best to have a volume control for the remote audio feeds coming back into the system