Proceedings of the 17th Linux Audio Conference (LAC-19), CCRMA, Stanford University, USA, March 23–26, 2019

JACKTRIP ON RASPBERRY PI

Chris Chafe, Scott Oshiro

Center for Computer Research in Music and Acoustics (CCRMA)
Stanford University, USA
cc@ccrma.stanford.edu soshiro@ccrma.stanford.edu

ABSTRACT

The jacktrip application for wide area network music performance has been ported to Raspberry Pi. The present setup runs Fedora 29 with the xfce desktop on a Model 3 B+ in conjunction with standard, low-cost stereo USB soundcards. We describe all the steps from initial OS installation through building and running jacktrip.

1. INTRODUCTION

Early study of internet acoustics at CCRMA[1] required the development of a system for low-latency, uncompressed audio streaming over IP. That software evolved into jacktrip[2] and is shared today as an open-source application widely used for jamming, rehearsing and concerts. Similar systems are discussed in a comprehensive review of network music performance technologies in [3].

The JamBerry project [4] (2014) was the first "Stand-Alone Device for Networked Music Performance Based on the Raspberry Pi" and like the present system was also linux-based. JamBerry provided uncompressed UDP audio streams and was in many ways an open-source equivalent to a proprietary solution existing at that time called jamLink[5]. Both involved custom software written for a system-on-a-chip (SOC) combined with an I^2S audio interface and both supported up to 4 peer-to-peer connections which were downmixed to stereo on the receiving side.

JamBerry included adaptive queuing, an automatic solution for tuning buffer size latency according to network conditions. The feature assisted musicians who would otherwise need to manually tune the parameter (in jacktrip's case, setting it's "-q" queue length parameter). Its user interface (UI) was presented on a built-in touchscreen and could also be operated via a remotetly-connected device (PC, mobile, tablet, etc.). An additional JamBerry feature is an error (packet loss) concealment procedure (reportedly an implementation of the algorithm used in the Opus Codec[6]).

The present project is simply a demonstration of running the standard release of jacktrip on generic, low-cost hardware, identical in all ways to running it on a linux-based PC (and functionally equivalent to running it on macOS or windows 10). The project differs from JamBerry on the audio and UI sides – it uses off-the-shelf USB sound cards, has less sophisticated packet loss concealment and no UI. The basic mode of operation provides a stripped-down, peer-to-peer bidirectional streaming engine with its usual command line incantations.

I/O ports on the the Raspberry Pi 3 Model B+ (rp3B+)[1] make it well-matched for the project. The built-in ethernet, HDMI and multiple USB ports make it simple to connect wired ethernet, display, mouse, keyboard and soundcard.

[1] `https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/`

The following is a full guide for those wishing to try jacktrip on the rp3B+. We also detail a musical demo of a more complex topolgy than simple peer-to-peer, an experiment involving multiple devices connected in a real-time waveguide mesh.

2. HOWTO

Before digging into installation details in the following sections, the first item we'll present is how to run jacktrip on an already prepared rp3B+.

Running is easy, it's equivalent to running jacktrip on more expensive hardware. But installing and building are different – these heavier operations are relatively sluggish and you need to plan for the time it takes. The approach detailed here uses the rp3B+ for all steps including a native compile. Fortunately, it's extremely easy to replicate the finished work simply by copying the image of a fully-prepared system for use on another rp3B+ device. How to make that happen will be explained, as well.

2.1. Running jacktrip on a fully-prepared rp3B+ (it's just the usual)

The application runs as described elsewhere[7] – nothing special is required. First, start `qjackctl`, a GUI-based application for setting up and starting the local jack audio server. Click on "Setup" and set the following: (example choices shown)

- select the desired (soundcard) Interface (M-Audio M-Track)
- Sample Rate (48000)
- Frames/Period (256)
- Periods/Buffer (3)

Click on "Ok" and click on "Start." The GUI's real-time status should start updating.

Next up, start jacktrip. Presumably, jacktrip has been installed system-wide (the last step in building jacktrip 4.1). You can check that it is there by opening a terminal and typing:

```
jacktrip -v
```

The result should show the current version.

As always, for this rp3B+ and each host it will connect to, the desired UDP port must be open for incoming traffic (the default port is 4464). This caveat is nicely satisfied in our Fedora environment by opening the `Firewall` application (requires administrator privileges), selecting the wired ethernet interface and setting its zone to `FedoraWorkstation`. See 4.2 for details.

To run your host as a jacktrip server, type:

```
jacktrip -s
```

Or, to run as client, type:

```
jacktrip -c <ipaddr>
```

where <ipaddr> corresponds to another host which is the jacktrip server you'll connect to. Type <ctrl>c to exit.

This howto now continues this with step-by-setp instructions for starting from scratch: installing the OS and build tools, obtaining source code, building and installing it, and configuring the environment.

3. INSTALLING FEDORA 29

We used 32G microSD cards for our demo and recommend working with the same. (As of this writing, a 32G microSD is now less expensive than a 16G microSD card.) Here are the steps we'll use to install a current version of the Fedora Linux operating system with the xfce desktop environment. Briefly,

- (laptop) download the image for arm7
- (laptop) flash it to a microSD card (18')
- (rp3B+) insert the microSD card, then power up the device
- (rp3B+) resize the image partition

The first two steps are accomplished on another computer, for example, a laptop and the next two are done on an rp3B+ which has been connected with wired ethernet, display, mouse and keyboard.

3.1. download the image for arm7 (using another computer)

(1.2G – about 8 minutes on a commodity ISP)

The base image we'll start with is a "spin" of Fedora 29 for armhf (ARM hard float) that's been pre-loaded with the xfce desktop environment. Download the compressed image here:
`https://download.fedoraproject.org/pub/fedora/linux/releases/29/Spins/armhfp/images/Fedora-Xfce-armhfp-29-1.2-sda.raw.xz`

3.2. flash it to a microSD card (using another computer)

(about 18 minutes on an i7 laptop)

Use the GUI application Etcher[2] to flash the image to a microSD card. Insert the microSD card into a laptop USB port using an adapter, launch Etcher, select the .xz file and select the 32G card. Flashing requires administrator privileges. When done, the adapter and microSD card can be removed.

3.3. insert flash, boot the rp3B+

(about 6 minutes with initial configuration steps)

It's advised to never insert or remove a microSD card while the rp3B+ is powered on. Insert the card and then turn on the power to boot the system. The user setup screen will boot into the xfce dektop environment. Specify a root password and a username and password.

3.4. resize the 4.7G partition

The initial image is too small for our purposes, so the next step is to resize it using the `gnome-disks` application which can be launched from a terminal. Select the largest current partition (Partition 3) at 4.7G and resize it to the maximum allowed.

[2] `https://www.balena.io/etcher/`

3.5. install build tools, libraries, utilities and applications

(about 25 minutes)

Open this document on the rp3B+ and enter the following commands as root in a terminal via copy and paste:

```
dnf -y install qt5-devel
dnf -y groupinstall "C Development Tools and Libraries"
dnf -y groupinstall "Development Tools"
dnf -y install jack-audio-connection-kit-devel alsa-lib-devel
dnf -y install iperf qjackctl audacity
```

4. PREPARING JACKTRIP

Here's what's remaining in order to to fully prepare jacktrip.

4.1. download, build and install the latest version of jacktrip

(about 9 minutes)

The current version of jacktrip is 1.2 and can be downloaded from its repository
`https://cm-gitlab.stanford.edu/cc/jacktrip`

To build the project,

- navigate to your download directory and extract the .zip file
- open a terminal in the jacktrip/src directory
- issue the command `./build` which runs the build script
- and then `sudo make install` to install the executable system-wide

4.2. open the UDP port

As previously mentioned, the network interface needs the incoming UDP port(s) used by jacktrip to be opened. Run the command `firewall-config` (or select "Firewall" from the Applications : Administration menu) and under "Options" do "Change Zones of Connections" for the Connection, Wired connection 1 (eth0), to make it be "FedoraWorkstation" (and switch it to "Permanent," if so desired, using "Options : Runtime to Permanent").

4.3. configure the rest of the environment

Good stuff to have includes permanent ssh service, realtime privileges and allowing remote soundcard access so that operating the device remotely is more useful. In the following, insert the desired account name where <user> is indicated:

```
systemctl start sshd.service
systemctl enable sshd.service
groupadd realtime
gpasswd -a <user> realtime
gpasswd -a <user> audio
echo "@realtime - rtprio 99" > /etc/security/limits.d/99-realtime.conf
echo "@realtime - memlock unlimited" >> \
/etc/security/limits.d/99-realtime.conf
```

Also choose your desired power save and screen saver configuration (automatically blanked screens during a performance are irritating). The rp3B+ is now fully-prepared any can be given a try using the instructions in 2.1.

5. REPLICATING AND MAINTAINING

5.1. replicating the image to another microSD card

A fully-prepared rp3B+ microSD card can be copied identically to another microSD card. Insert the source microSD card in another computer, for example, a laptop using a USB adapter. Then, taking care to select the correct storage volume, do a "dd" disk copy from the source microSD card to the laptop's own storage. It's essential to check the volumes on the laptop via a terminal command

```
df -h
```

and determine by experiment which volume gets added / removed when the microSD card is plugged / unplugged. For example, a volume named `/dev/sdb1` may be correlated with this behavior indicating that a device `/dev/sdb` will be the source image (input file) we want to copy from. Choose a suitable output file path and name.

Again, beware! Realize that the following command is only an example. It refers to `/dev/sdb` and that may be incorrect if your particular setup at the time you launch the command has the microSD card showing as a different volume. For example, if you have another USB drive under that name and if it's really big, the operation could fill up your home directory (written from experience).

sudo dd if=/dev/sdb of=~/rp1.img status=progress

When done the output image file can then be flashed to a new microSD card using the Etcher application mentioned in 3.2 above. Select the output image (output file) from "dd" as the source.

6. TESTING RESULTS

The rp3B+ has been compared side-by-side with an i7 laptop using a method developed for determining the quality-of-service (QoS) under a jacktrip load. The host being tested connects to a loopback server [8] which echoes back the host's own audio. While so doing, the server records elapsed time between packets (inter-packet intervals) arriving from the test device. Ideally, these would be absolutely regular but in practice cannot be. Network and host characteristics affect the degree of deviation.

In Figure 1 the rp3B+ registered 2 inter-packet intervals exceeding the input buffer queue cushion shown by the black line and the laptop registered 1 exceeding this deadline (after approximately 25k packets). The test was conducted over a local network via the ethernet ports of a 5-year old commodity-grade router (D-Link DIR-655). Network QoS factors in other setups may mask the differences between the two devices tested but in this case fine-grained differences are apparent. If this test were run over a commodity wide area network like cable or over a local wifi connection it would clearly show the effects of congestion in the former and hardware in the latter. See [9] for a comparative "Evaluation of Network Music Technology on Public and Private Networks."

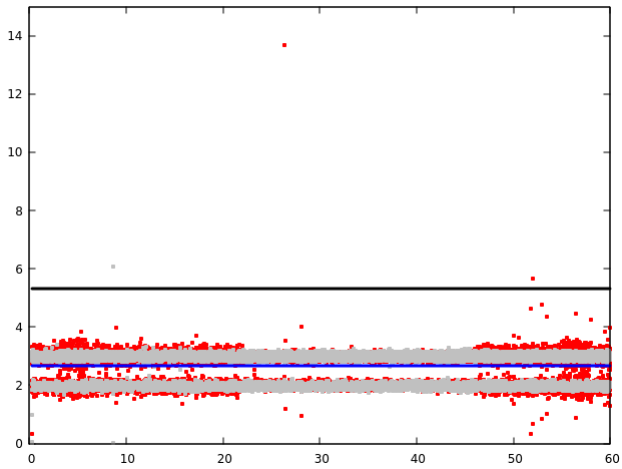

Figure 1: *A comparison of regularity of streams transmitted by an rp3B+ (red) and a Lenovo P51 i7 laptop (grey). The Y-axis shows the inter-packet intervals in ms of packets arriving at a jacktrip host which logged approximately 25k packets over 60 seconds. Nominal period (blue line) is $2.67ms$ (128 samples / packet at $48kHz$ sample rate). Black line corresponds to maximum input buffer latency (-q 2 is shown).*

7. THE LAC DEMO: A WAVEGUIDE MESH SYNTH USING RASPBERRY PI'S

An ensemble of Rapsberry Pi's was tethered in common to an isolated LAN. They were remotely operated via SSH login from the demo laptop which connected them to each other to create a streaming waveguide mesh configuration.

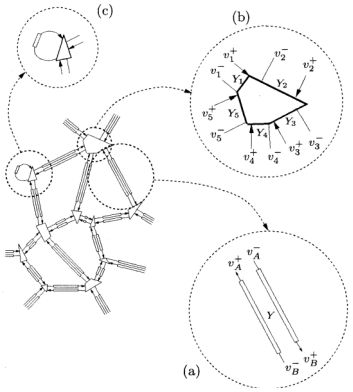

Figure 2: *Irregular waveguide meshes of arbitrary complexity can be made with network nodes running jacktrip. Reprinted from [10], "Portion of a digital waveguide network, and enlarged views of its principal components: (a) a bidirectional delay line, of delay duration T and admittance Y (accepting two waves v and v output from scattering junctions, delaying them, and producing two waves v and v each of which is then incident on a scattering junction); (b) a scattering junction connected to five waveguides of admittances Y ; ... ; Y , (accepting, in this case, five input waves v ; ... ; v , and yielding five output waves v ; ... ; v); and (c) a self-loop."*

The demo was inspired by the concept of irregular digital waveguide networks, such as shown in Figure 2 (reprinted from [10]). Arbitrary meshes can be created of nodes interconnected by bi-directional streams. The sound that results is a resonant object that when impulsed or otherwise excited rings in more or less complex ways depending on the topology chosen. The simplest 3-node structure consists of a central scattering node junction (a jacktrip server running -S) with two edge nodes connected to it (jacktrip clients running -C). Audio arriving at the edge nodes from the server is looped back to it and the server itself implements a 2-port scattering junction by summing both incoming streams, attentuating the result and sending that signal to the clients. At least one of the edges includes a low-pass filter in its loopback path. The nodes are operating in a hub-and-spoke topology described in [8].

8. REFERENCES

[1] Chris Chafe, Scott Wilson, and Daniel Walling, "Physical model synthesis with application to internet acoustics," in *Proc. 2002 Intl. Conference on Acoustics, Speech and Signal Processing*. 2002, pp. IV–4056–IV–4059, IEEE.

[2] Juan-Pablo Cáceres and Chris Chafe, "Jacktrip: Under the hood of an engine for network audio," *J. New Music Res.*, vol. 39, no. 3, pp. 183–187, 2010a.

[3] Cristina Rottondi, Chris Chafe, Claudio Allocchio, and Augusto Sarti, "An overview on networked music performance technologies," *IEEE Access*, vol. 4, pp. 8823–8843, 2016.

[4] Florian Meier Marco Fink and Udo Zoelzer, "The jamberry - a stand-alone device for networked music performance based on the raspberry pi," in *Linux Audio Conference*, May 2014 (accessed February 18, 2019), http://lac.linuxaudio.org/2014/papers/6.pdf.

[5] Scott Kahn, *MusicianLink jamLink*, 2012 (accessed February 18, 2019), `https://musicplayers.com/reviews/live_sound/2011/1111_jamLink.php`.

[6] Koen Vos Jean-Marc Valin and Timothy B. Terriberry, "Definition of the opus audio codec," *Internet Engineering Task Force*, vol. RFC 6716,, 2012.

[7] Chris Chafe, "I am streaming in a room," *Frontiers in Digital Humanities*, vol. 5, pp. 27, 2018.

[8] Juan-Pablo Cáceres and Chris Chafe, "Jacktrip/soundwire meets server farm," *Computer Music Journal*, vol. 34, no. 3, pp. 29–34, 2010b.

[9] Trevor Henthorn Chris Chafe and Sarah Weaver, "Evaluation of network music technology on public and private networks," in *NowNet Arts Conference*, 2018 (accessed February 18, 2019), `https://ccrma.stanford.edu/~cc/deck.js/nownet2018performanceData/`.

[10] Stefan Bilbao and III Julius O. Smith, "Finite difference schemes and digital waveguide networks for the wave equation: Stability, passivity, and numerical dispersion," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 3, pp. 255–266, 2003.