Trabajo - Instalación, Administración y Uso de un servicio en Linux

Compartición de recursos, Audio

Servicios Telemáticos

Departamento de Ingeniería Telemática

IMPORTANTE: **Antes** de rellenar esta plantilla **lea detenidamente** los documentos "**Enunciado** del Trabajo" y "**FAQ** del Trabajo".

El contenido esperado para cada uno de los apartados de esta memoria se indica en el **Anexo A** del documento "**Enunciado** del Trabajo".

Juan Castelo Rus

o

ÍNDICE

Herramientas analizadas	Asimiladas a
JACK2	***Servidor***
CHUCK y Pure Data	***Cliente***

1. Objetivos y Alcance

1.1. Introducción

El mundo de la música está muy ligado a la tecnología actual, ya que la necesidad de producir música, de manera eficiente, la grabación sea un proceso más a meno o el auge de sintetizadores musicales, por tanto, es necesario crear servicios que permitan gestionar de una manera eficiente los distintos sonidos producidos por los artistas, ya sean sus voces, instrumentos musicales o mesas de mezclas. También está el crecimiento del uso de programas de programación musical para el aprendizaje y creación de sonidos en tiempo real.

Todo lo contado anteriormente se acrecentó aún más, tras la pandemia mundial del Covid-19, donde los músicos tuvieron que utilizar sus propios recursos para poder generar música o el crecimiento de plataformas virtuales para enseñar música.

1.2. Motivación y funcionalidad del servicio

JACK2, nos permite la posibilidad desde una interfaz gráfica poder redireccionar tanto las entradas como las salidas de audios que se generen en nuestro sistema, además permite que múltiples aplicaciones estén generando sonido simultáneamente. Otra característica de JACK2, es que permite, la conexión remota, es decir en un red local se puede crear una estructura master/slave, donde un ordenador va a ser el servidor maestros y uno o más ordenadores van a ser los esclavos. Esto permite que si por ejemplo llega una entrada de sonido al maestro se pueda redirigir a un esclavo, donde se sintetice ese sonido y tras sintetizar dicho sonido devolver al maestro.

Chuck y Pure Data, son dos lenguajes de programación y procesadores de audio. El primero es un lenguaje de programación creado por la universidad de Princeton para la

creación y aprendizaje musical a través de los ordenadores. El segundo lenguaje de programación, que es Pure Data, es un sintetizador, expandido ya que es un lenguaje visual no hay que escribir código, sino que se realiza a base de bloques. Por tanto ambos programas son muy buenas herramientas para el aprendizaje musical.

1.3. Documentación bibliográfica

Texto. Recuerde, lista de referencias con el formato:

JACK2:

[1] https://github.com/jackaudio/jackaudio.github.com/wiki/WalkThrough_User_NetJack2

[2] https://wiki.archlinux.org/title/JACK_Audio_Connection_Kit_(Espa%C3%B1ol)#Configuraci%C3%B3n_b%C3%A1sica

[3] https://github.com/jackaudio/jackaudio.github.com/wiki

[4] https://github.com/jackaudio/jack2

Chuck:

[5] https://www.youtube.com/watch?v=toFvb6uqiDc&list=PL-9SSIBe1phI_r3JsylOZXZyAXuEKRJOS

[6] https://chuck.stanford.edu/doc/learn/tutorial.html

[7] https://chuck.cs.princeton.edu

Pure Data:

[8] https://puredata.info

[9] https://github.com/pure-data/pure-data

[10] https://supercollider.github.io

[11] https://csound.com

2. Base Teórica

En este apartado se realizará un breve resumen sobre los conceptos fundamentales del servicio

2.1. Descripción del servicio y conceptos implicados

Servidor de audio: Un servidor de audio es un sistema o programa informático diseñado para gestionar la reproducción, grabación y procesamiento de señales de audio. Su función principal es coordinar y distribuir datos de audio entre diferentes aplicaciones y dispositivos. Los servidores de audio son esenciales en entornos donde se requiere un control preciso sobre la reproducción de audio, como en la producción musical o la edición de sonido.

Lenguaje de programación visual: Un lenguaje de programación visual utiliza representaciones gráficas en lugar de texto para expresar algoritmos o lógica de programación. En lugar de escribir líneas de código, los programadores utilizan bloques, iconos o diagramas para construir el flujo de su programa.

Lenguaje de Programación de Audio: Un lenguaje de programación de audio se centra específicamente en la manipulación y procesamiento de datos de audio.

Sintetizador: Un sintetizador es un dispositivo electrónico o un software que crea sonidos artificialmente. Puede generar una amplia variedad de sonidos, desde recreaciones realistas de instrumentos musicales hasta sonidos completamente originales y abstractos. Los sintetizadores trabajan manipulando ondas sonoras básicas, como ondas sinusoidales, para producir tonos complejos. Pueden ser analógicos o digitales y se utilizan comúnmente en la música electrónica y en la producción musical en general.

2.2. Protocolos utilizados por el servicio

A continuación, se van a comentar los protocolos usados en el servicio.

2.2.1. Protocolos Comunes

Protocolos	RFC
MDNS	6762
UDP	768
IGMP	1112
ICMP	792

Tabla 1 Protocolos generales

Estos son los protocolos que se van a utilizar en el servicio de JACK2 y a continuación, una explicación breve de cada uno:

1. MDNS: es un protocolo de resolución de nombres que permite a los dispositivos en una red local descubrir y comunicarse entre sí utilizando nombres de host sin la necesidad de un servidor de nombres de dominio (DNS) centralizado. Funciona a través de multicast en UDP y es utilizado comúnmente para servicios de impresión, descubrimiento de dispositivos y otras aplicaciones locales

2. UDP: es un protocolo de transporte ligero y sin conexión. A diferencia de TCP, no establece una conexión antes de enviar datos y no garantiza la entrega ordenada de paquetes ni la detección de errores. UDP es utilizado cuando la velocidad y la eficiencia son más críticas que la integridad de los datos, como en transmisiones de audio y video, juegos en línea y servicios de transmisión en tiempo real.

3. IGMP: es un protocolo utilizado para la administración de grupos de multidifusión en redes IP. Permite a los hosts informar a los enrutadores sobre su deseo de unirse a o abandonar un grupo de multidifusión. IGMP es esencial para la transmisión de datos a múltiples receptores en una red, como en IPTV o transmisiones de video en vivo.

4. ICMP: es un protocolo de control de mensajes utilizado por dispositivos de red para enviar mensajes de error y control. Es esencialmente parte de la capa de red y se utiliza para informar sobre problemas de red, como mensajes de "ping" para verificar la accesibilidad de un host y mensajes de error cuando se encuentra un problema en la entrega de paquetes IP.

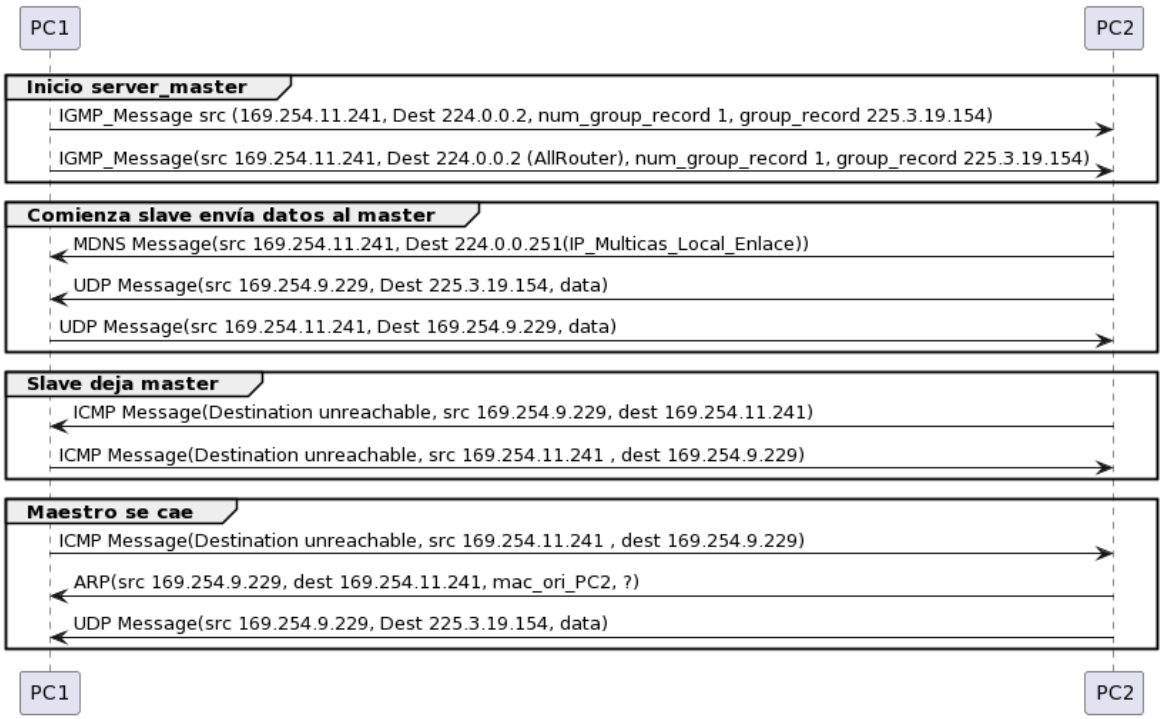


Ilustración 1 Esquema diagrama de paso de mensaje.

Este es el diagrama de pasos de mensaje que se va a realizar en el proyecto.

3. Evaluación de la implementación estudiada

Se realizará un análisis comparativo entre la implementación del servicio que se usará en el trabajo y las demás implementaciones encontradas en el mercado.

3.1. Descripción y características de la implementación de servicio estudiada

Se van a describir el servicio de servidor de audio JACK2 y las aplicaciones de programación Chuck y Pure Data.

3.1.1. Breve Descripción de la solución adoptada

Servicio	JACK2
Funcionalidad	Este servicio desde una interfaz gráfica nos permite poder redireccionar tanto las entradas como las salidas de audios que se generen en nuestro sistema. Además, también nos permite realizar un conexión en una red de área local master/slave entre diversos servidores JACK2.
Documentación bibliográfica	Las referencias usadas son desde la [1] a la [6]
Equipo Software	Equipo de desarrollo de JACK
Versión	JACK 1.9.22-02/02/2023
Última Versión disponible	JACK 1.9.22-02/02/2023

Tabla 2 Descripción de JACK2

3.1.2. Equipamiento necesario

Servicio	JACK2
Hardware	Un ordenador
Red	Diversos ordenadores conectados a la misma red local
Elemento Software	Descargar de GitHub la última versión [6] y añadir en el ordenador la posibilidad de funcionamiento real y que el usuario dit tenga los permisos para ello.
Sistema Operativo	Ubuntu 22.04

Tabla 3 Equipamiento del servidor JACK2

3.1.3. Características y funcionalidades

JACK2:

- Audio de Baja Latencia: Jack2 destaca por su capacidad para proporcionar audio de baja latencia, lo que lo hace ideal para aplicaciones en tiempo real como la producción musical y la grabación.

- Conectividad Flexible: Permite la interconexión de múltiples aplicaciones de audio y hardware de manera eficiente, facilitando la colaboración y el flujo de trabajo en entornos complejos.

- Compatibilidad Multiplataforma: Jack2 es compatible con varios sistemas operativos, incluyendo Linux, macOS y Windows, lo que lo convierte en una solución versátil para una amplia gama de usuarios.

- Canalización de Audio: Ofrece una potente capacidad de enrutamiento de audio, permitiendo a los usuarios dirigir señales de audio entre diferentes aplicaciones y dispositivos con gran flexibilidad.

3.2. Comparativa de soluciones existentes en el mercado

Se va a usar JACK2, ya que es un implementación que nos permite realizar conexiones remotas entre diversos ordenadores, por su baja latencia que es importante para las aplicaciones en tiempo real y también por la interfaz gráfica que nos permite monitorizar las entradas y salidas de audio de nuestro ordenador.

Servicios	PULSEAUDIO	PIPEWIRE	JACK2
Versión	15.99.1	0.3	1.9.22
Sistema Operativo	Linux macOS Windows BSD	Linux	Linux macOS Windows
Equipo Software	Proyecto PulseAudio	Equipo de Desarrollo de PipeWire	Equipo de Desarrollo de JACK
Licencia	LGPL	LGPL	GPL
Lenguaje de programación	C	C	C,C++
Dependencias	init-system-helpers, libasound2,..	libfdk-aac2 libldacbt-{abr,enc}2 libopenaptx0	ALSA, portaudio y libasound2,..
Diferencias	Gestión de sonido y mezcla	Unificación de ambos servicios tanto de JACK2 como PULSEAUDIO	Baja latencias en el servicio y aplicaciones audio
Nivel de uso	Popularmente usado	En proceso de sustitución tanto de JACK2 como PulseAudio aunque no está muy expandido	Popularmente usado para aplicaciones de audios y administración de estas

Tabla 4 Comparativa entre diversos servicios

3.3. Clientes para el servicio

Chuck:

- Programación Musical en Tiempo Real: Chuck es un lenguaje de programación específico para música que permite la programación en tiempo real, facilitando la creación y manipulación de sonidos de manera dinámica.
- Síntesis de Sonido: Chuck está diseñado para la síntesis de sonido y el procesamiento de señales de audio en tiempo real. Ofrece abstracciones para la creación de instrumentos y la manipulación de eventos musicales.
- Orientado a Eventos: Su modelo de programación está orientado a eventos, lo que significa que los sonidos y las acciones se pueden controlar y modificar en respuesta a eventos específicos.
- Multiplataforma: Chuck es multiplataforma y puede ejecutarse en sistemas operativos como Linux, macOS y Windows.

Pure Data:

- Entorno de Programación Visual: Pure Data utiliza un entorno de programación visual que permite a los usuarios crear y manipular procesos de sonido mediante la conexión de objetos gráficos, facilitando la creación de patches visuales.
- Procesamiento de Señales Multimedia: Diseñado para el procesamiento de señales multimedia, Pure Data se utiliza para la creación de música electrónica, instalaciones sonoras interactivas y más.
- Comunidad Activa: Pure Data cuenta con una comunidad activa de usuarios y desarrolladores que contribuyen con bibliotecas y recursos, lo que facilita la expansión de sus funcionalidades.
- Multiplataforma: Es compatible con varios sistemas operativos, lo que permite a los usuarios utilizarlo en entornos que van desde Linux hasta Windows y macOS.

3.3.1. Referencias y características del cliente adoptado

Servicio	Chuck
Funcionalidad	Lenguaje de programación musical, diseñado para el aprendizaje y educación en el manejo de sintetizadores de audio y creación de música electrónica.
Documentación bibliográfica	Las referencias usadas son desde la [5] a la [7]
Equipo Software	Universidad de Princeton
Versión	CHUCK 1.5.1.8-10/2023
Última Versión disponible	CHUCK 1.5.1.8-10/2023

Tabla 5 Características de Chuck

Servicio	Pure Data
Funcionalidad	Sintetizador de audio, y el cual es muy intuitivo y sencillo de manejar al ser un lenguaje de programación visual.
Documentación bibliográfica	Las referencias usadas son desde la [8] a la [10]
Equipo Software	Equipo de desarrollo de Miller S. Puckette's "vanilla"
Versión	JACK 0.54.1-31/10/2023
Última Versión disponible	JACK 0.54.1-31/10/2023

Tabla 6 Características de Pure Data

3.3.2. Comparativa de clientes existentes en el mercado

Servicios	SuperCollider	Csound	Pure Data	Chuck
Versión	3.13.0-20/02/2023	6.18.1	0.54-1	1.5.1.8
Sistema Operativo	Linux macOS Windows	Linux macOS Windows	Linux macOS Windows	Linux macOS Windows
Licencia	GPL	GPL	BSD	GLP
Lenguaje de programación	Principalmente C++, C	C	Principalmente C++, C	C++
Diferencias	Música experimental y diseño sonoro	Muchos años en síntesis y procesamiento de sonido	Programación visual y multimedia	Diseñado para música y sonido en tiempo real
URL	[11]	[12]	[8]	[7]

Tabla 7 Tabla comparativa entre clientes

4. Proceso de instalación y Uso del cliente

A continuación, se van a mostrar los pasos de instalación de ambos clientes.

4.1. Obtención del software del cliente

4.1.1 Chuck

1. Descargar de la referencia [9], el archivo Chuck-1.5.8.1.tgz
2. Realizar el comando `tar -xvzf chuck-1.5.1.8.tgz`

4.1.2 Pure Data

1. Realizar el comando `git clone https://github.com/pure-data/pure-data.git`

4.2. Instalación del cliente

A continuación, se explica la instalación de ambos clientes.

4.2.1. Primera instalación del cliente

4.2.1.1 Chuck

Los pasos para instalar el cliente Chuck son los siguientes:

1. Descomprimir el archivo con el comando `tar -xvzf chuck-1.5.1.8.tgz`
2. Ingresar en la carpeta Chuck-1.5.1.8 `cd chuck-1.5.1.8`
3. Instalar dependencias: `sudo apt install -y build-essential bison flex libsndfile1-dev libasound2-dev libpulse-dev libjack-jackd2-dev`
4. Ingresar en src: `cd src`
5. Compilar Linux: `make Linux`
6. Comprobar instalación con: `./chuck --version`

4.2.1.2 Pure Data

1. Lo primero será comprobar que las dependencias están instaladas: `sudo apt-get install build-essential tcl tk libasound2-dev`
2. Clonar repositorio: `git clone https://github.com/pure-data/pure-data.git`
3. Entrar en la carpeta del repositorio: `cd pure-data`
4. Configurar Pure Data:
    a. `./autogen.sh`
    b. `./configure --enable-jack`
5. Instalar
    a. `make`
    b. `make install`
6. Comprobar instalación: `pd --version`

4.2.2. Desinstalación del cliente

4.2.2.1 Chuck

Para desinstalar Chuck hay que ejecutar el comando `sudo apt-get remove chuck`

4.2.2.2 Pure Data

Para desinstalar Chuck hay que ejecutar el comando `sudo apt-get remove puredata`

4.3. Configuración del cliente

4.3.1 Chuck

Para que Chuck pueda conectarse con el servidor JACK2, habrá que usar el comando: `./chuck --driver:jack archivo.ck`

4.3.2 Pure Data

En el caso de Pure Data habrá que ejecutar el comando: `./pd -jack`

5. Proceso de instalación/administración del servidor

A continuación, se comentará el proceso de instalación del servidor.

5.1. Obtención del software del servidor

El servidor se obtiene ejecutando el comando siguiente: `git clone https://github.com/jackaudio/jack2`

5.2. Instalación del servidor

5.2.1. Primera instalación del servicio

5.2.1.1. Instalación desde código fuente (Compilación)

Para la instalación hay que seguir los siguientes pasos:

1. Instalar las dependencias necesarias: `sudo apt-get install -y libopus-dev portaudio19-dev libasound2-dev libffado-dev libgtkmm-2.4-dev libeigen3-dev`
2. Entrar en la carpeta de jack2: `cd jack2`
3. Cambiar la versión de Python a Python3 del archivo waf: `nano/code`
4. Configurar e instalar JACK:
    a. `./waf configure`
    b. `./waf build`
    c. `sudo ./waf install`
5. También, habilito a JACK2 el soporte de ASLA y PortAudio:
    a. `echo "driver alsa" | sudo tee -a ~/.jackdrc`
    b. `echo "driver real_time 1" | sudo tee -a ~/.jackdrc`

```
c. echo "driver seq" | sudo tee -a ~/.jackdrc

d. echo "realtime-priority 99" | sudo tee -a ~/.jackdrc

e. echo "realtime" | sudo tee -a ~/.jackdrc

f. echo "softmode" | sudo tee -a ~/.jackdr
```

6. Configuro archivos para permitir el tiempo real:

```
a. echo "@audio - rtprio 99" | sudo tee -a
   /etc/security/limits.conf

b. echo "@audio - memlock unlimited" | sudo tee -a
   /etc/security/limits.conf
```

7. Configuro el Kernel:

```
a. sudo echo "snd-aloop" | sudo tee -a /etc/modules

b. echo "options snd-aloop index=0 pcm_substreams=2" | sudo tee
   -a /etc/modprobe.d>

c. sudo update-initramfs
```

5.2.1.2. Instalación desde paquetes/repositorios

Para instalar por paquetes hay que hacer los siguientes pasos:

- `apt-get install jack2`

5.2.2. Actualización del servicio

5.2.2.1. Actualización desde código fuente

Para realizar la actualización por código fuente habrá que seguir los pasos explicados anteriormente en la instalación por código fuente.

5.2.2.2. Actualización desde paquetes/repositorios

Para realizar la actualización por paquete se puede hacer de dos maneras o desinstalas la versión anterior y después vuelves a instalarla como se explica en el apartado 5.2.1.2 o puedes ejecutar el comando `apt update` y después `apt upgrade`.

5.2.3. Desinstalación del servicio

5.2.3.1. Desinstalación desde código fuente

Para la desinstalación por código fuente hay que hacer lo siguiente:

1. Borrar dependencias: `sudo apt-get remove --purge -y libopus-dev portaudio19-dev libasound2-dev libffado-dev libgtkmm-2.4-dev libeigen3-dev`
2. Desinstalar JACK: `./waf uninstall`
3. Borrar archivos de instalación: `sudo rm -rf jack2`
4. Deshacer los cambios anteriores en la configuración del kernel:
   a. `sudo sed -i '/@audio - rtprio 99/d' /etc/security/limits.conf`
   b. `sudo sed -i '/@audio - memlock unlimited/d' /etc/security/limits.conf`
   c. `sudo sed -i '/snd-aloop/d' /etc/modules`
   d. `sudo rm -f /etc/modprobe.d/alsa-loopback.conf`
   e. `sudo update-initramfs -u`
5. Resetear el ordenador: `reboot`

5.2.3.2. Desinstalación desde paquetes/repositorios

Para desinstalar por paquetes JACK2, se realiza de la siguiente manera:

1. `apt-get remove jack2`
2. `apt-get purge jack2`

5.3. Configuración del servidor

La configuración del servidor se realiza a través de la interfaz gráfica del servidor, que es donde se pueden modificar los distintos parámetros que nos ofrece el servicio como la frecuencias de obtención de datos, número de canales, etc. Aunque también, se puede hacer desde la línea de comandos.

Para el uso del real time en el ordenador hay que modificar el fichero limits.conf, que se encuentra en la carpeta etc/security y hay que añadirle los siguientes parámetros:

- `echo "@audio - rtprio 99" | sudo tee -a /etc/security/limits.conf`
- `echo "@audio - memlock unlimited" | sudo tee -a /etc/security/limits.conf`

También, habilito a JACK2 el soporte de ASLA y PortAudio, modificando el archivo jackdrc.

Por último, configuro el KERNEL, para permitir el acceso a tiempo real, para esto modifico el etc/modules para añadir snd-aloop y también modifico el fihcero /etc/modprobe.d.

6. Puesta en funcionamiento del servicio

Se van a detallar los pasos a seguir para arrancar el servicio

6.1. Arranque del servicio con el sistema

Haciendo uso del arranque Systemd, habrá que seguir los siguientes pasos

- `su root`
- `nano /etc/systemd/system/jack.service`, generamos el script con la siguiente información:

  [Unit]

  ```
  Description=Jack Audio Connection Kit
  ```

  [Service]

  ```
  Type=simple

  Environment="JACK_NO_AUDIO_RESERVATION=1"

  ExecStart=/usr/bin/jackd -d alsa
  ```

  [Install]

  ```
  WantedBy=default.targe
  ```

- `sudo systemctl daemon-reload`
- `sudo systemctl enable jack.service`
- `sudo systemctl start jack.service`

6.2. Administración y monitorización del funcionamiento

6.2.1. Configuración y Uso de los ficheros de registro

No procede, la única manera que se me ocurre es usando el modo verbose y mandando la salida de este modo a un archivo log.

6.2.2. Arranque del servicio en modo detallado (verbose)

Para poder arrancar el servicio en modo verbose se realiza de la siguiente manera: j`ackd --v d alsa`. La diferencia entre los modos `-v, -vv y -vvv,` es que este último aporta más información que los dos primeros, que `-vv` nos proporciona más información que el primero y, pero menor que el tercero y el primero solo nos aporta información básica del servicio.

6.2.3. Seguridad del servicio

No procede.

6.3. Tests y Pruebas del correcto arranque del servicio

```
dit@linux-U-L1:~$ jackd -v -d alsa
jackdmp 1.9.22
Copyright 2001-2005 Paul Davis and others.
Copyright 2004-2016 Grame.
Copyright 2016-2023 Filipe Coelho.
jackdmp comes with ABSOLUTELY NO WARRANTY
This is free software, and you are welcome to redistribute it
under certain conditions; see the file COPYING for details
no message buffer overruns
no message buffer overruns
no message buffer overruns
JACK server starting in realtime mode with priority 10
self-connect-mode is "Don't restrict self connect requests"
Jack: JackPosixThread::StartImp : create non RT thread
Jack: JackPosixThread::ThreadHandler : start
Jack: JackDriver::Open capture_driver_name = hw:0
Jack: JackDriver::Open playback_driver_name = hw:0
Jack: Check protocol client = 9 server = 9
Jack: JackEngine::ClientInternalOpen: name = system
Jack: JackEngine::AllocateRefNum ref = 0
Jack: JackLinuxFutex::Allocate name = jack_sem.1002_default_system val = 0
Jack: JackEngine::NotifyAddClient: name = system
Jack: JackGraphManager::SetBufferSize size = 1024
Jack: JackConnectionManager::DirectConnect first: ref1 = 0 ref2 = 0
Jack: JackGraphManager::ConnectRefNum cur_index = 0 ref1 = 0 ref2 = 0
Jack: JackDriver::SetupDriverSync driver sem in flush mode
creating alsa driver ... hw:0|hw:0|1024|2|48000|0|0|nomon|swmeter|-|32bit
configuring for 48000Hz, period = 1024 frames (21.3 ms), buffer = 2 periods
ALSA: final selected sample format for capture: 32bit float little-endian
ALSA: use 2 periods for capture
ALSA: final selected sample format for playback: 32bit float little-endian
ALSA: use 2 periods for playback
Jack: JackSocketServerChannel::Open
Jack: JackServerSocket::Bind : addr.sun_path /dev/shm/jack_default_1002_0
Jack: JackSocketServerChannel::BuildPoolTable size = 1
Jack: JackEngine::Open
Jack: JackClientSocket::Connect : addr.sun_path /dev/shm/jack_default_1002_0
```

Ilustración 2 Demostración del servicio arranca en modo verbose

En esta imagen se puede observar cual es comando para iniciar el servidor en modo verbose y algunas salida de este cuando esta en este modo.

```
root@linux-U-L1:/etc/systemd/system# sudo systemctl status jack.service
● jack.service - Jack Audio Connection Kit
     Loaded: loaded (/etc/systemd/system/jack.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2023-12-06 20:03:01 CET; 2s ago
   Main PID: 51269 (jackd)
      Tasks: 4 (limit: 4608)
     Memory: 122.8M
        CPU: 173ms
     CGroup: /system.slice/jack.service
             └─51269 /usr/bin/jackd -d alsa hw:0

dic 06 20:03:01 linux jackd[51269]: This is free software, and you are welcome to redist
dic 06 20:03:01 linux jackd[51269]: under certain conditions; see the file COPYING for c
dic 06 20:03:01 linux jackd[51269]: JACK server starting in realtime mode with priority
dic 06 20:03:01 linux jackd[51269]: self-connect-mode is "Don't restrict self connect re
dic 06 20:03:01 linux jackd[51269]: creating alsa driver ... hw:0|hw:0|1024|2|48000|0|0|
dic 06 20:03:01 linux jackd[51269]: configuring for 48000Hz, period = 1024 frames (21.3
dic 06 20:03:01 linux jackd[51269]: ALSA: final selected sample format for capture: 32bi
dic 06 20:03:01 linux jackd[51269]: ALSA: use 2 periods for capture
dic 06 20:03:01 linux jackd[51269]: ALSA: final selected sample format for playback: 32b
dic 06 20:03:01 linux jackd[51269]: ALSA: use 2 periods for playback
```

Ilustración 3 Demostración de inicio de JACK2 arranque ordenador

En esta imagen se observar el estado del servidor después de haber realizado las modificaciones en el sistema para que se inicie cuando arranca el sistema.

7. Diseño de los escenarios de prueba

7.1. Escenario de Defensa 1: Demostración de compartición de audio de manera local

En este escenario se va a demostrar uno de los usos más frecuentes, que he encontrado para el servidor JACK2. Consiste en lo siguiente se tienen una o más entradas de audios en el sistema, ya sea a través de MIDI, o de un sintetizador de sonido, entonces haciendo uso de la interfaz gráfica de JACK2, que se llama qjackctl, se pueden redirigir estas entradas de audios de diversas maneras, como por ejemplo que salgan por la tarjeta de sonido de nuestro sistema o redirigirlas a otra aplicación para poder realizar diversas modificaciones al sonido generado en la entrada, es una funcionalidad muy importante he creo necesaria realizar un escenario con esta configuración. Se va a utilizar el driver ALSA.

7.1.1. Escenario 1: Esquema del escenario

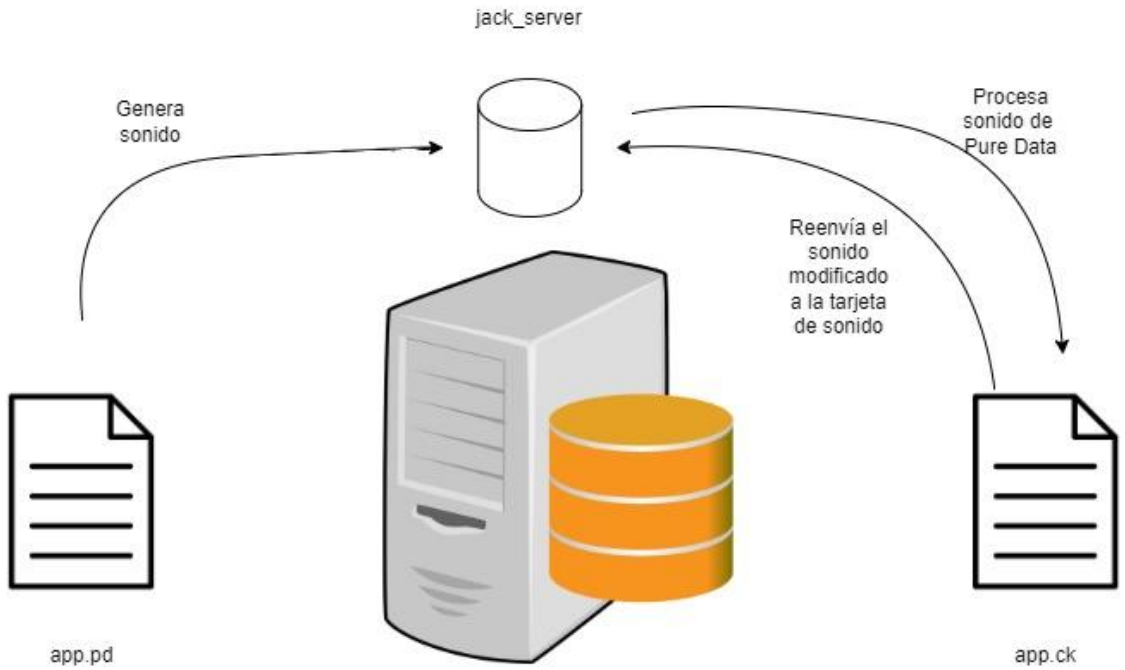


Ilustración 4 Esquema escenario 1

Este es el esquema del primer escenario, donde tenemos una aplicación en Pure Data que genera un sonido el cual lo envía al servidor local JACK2, en este servidor y haciendo uso de la herramienta qjackctl, redireccionamos la salida de Pure Data al programa Chuck, donde se van a realizar diversas modificaciones, al sonido entrante para poder devolverlo al servidor JACK2 y que redireccione la salida de Chuck a la tarjeta de sonido del sistema.

7.1.2. Escenario 1: Configuración del servidor

Para la configuración del servidor, hay que seguir los pasos de instalación, explicados en el apartado 5.2.1.1, una vez instalado el servidor y la herramienta gráfica qjackctl, solo habrá que abrir qjackctl y seleccionar la interfaz de salida del servidor JACK2.

7.1.3. Escenario 1: Tests y Pruebas del Escenario

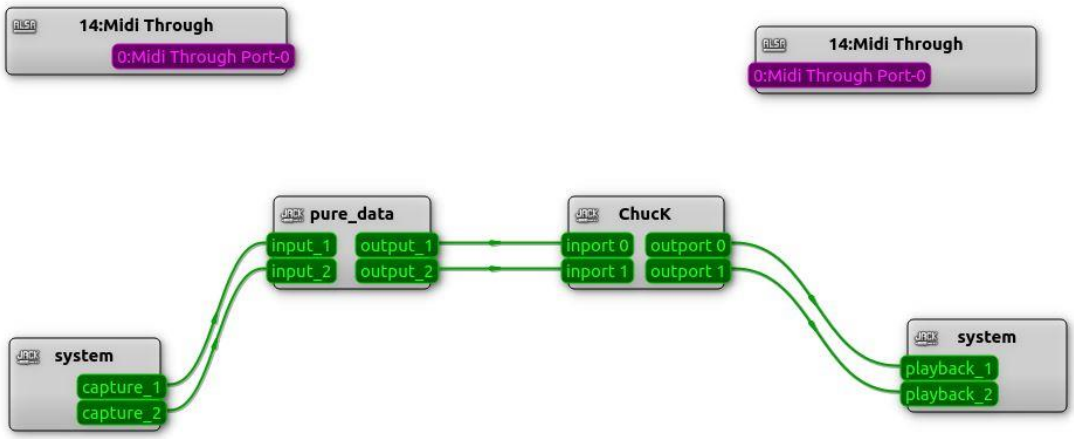


Ilustración 5 Escenario_1

Aquí en esta imagen se observa la interfaz gráfica de qjackctl y la conexión para poder obtener la salida de audio del sistema. En la imagen de abajo se puede observar la salida de logs de qjackctl cuando Pure Data emite un sonido y luego lo transmite Chuck.

Para poder realizar esta prueba se han hecho uso del comando qjackctl para poder abrir la interfaz gráfica. Después se ha ejecutado el comando ./chuck –driver:Jack ~/Descargas/Trabajo_STA/test_prueba_1.ck, este nos ejecuta el programa de chuck para poder recibir el sonido de Pure Data y enviarlo de nuevo a Jack y, por último, se ha ejecutado el comando ./pd -jack para poder iniciar el sintetizador de Pure Data.

```
17:25:20.780 XRUN callback (354).
17:25:22.307 XRUN callback (2 omitidos).
17:25:48.518 XRUN callback (357).
17:25:50.549 XRUN callback (8 omitidos).
17:25:51.086 XRUN callback (366).
17:25:52.192 XRUN callback (367).
17:25:52.558 XRUN callback (2 omitidos).
17:26:22.022 XRUN callback (369).
17:26:22.625 XRUN callback (3 omitidos).
17:26:33.443 XRUN callback (373).
17:26:33.628 XRUN callback (1 omitidos).
17:26:35.640 XRUN callback (5 omitidos).
17:26:35.799 XRUN callback (381).
17:26:37.682 XRUN callback (3 omitidos).
```

Ilustración 6 Salida de qjackctl

7.1.4. Escenario 1: Análisis del intercambio real de mensajes de red

No procede

7.2. Escenario de Defensa 2: Uso de JACK2 remotamente

En este escenario se va a comprobar y demostrar el uso de JACK2, para su uso de manera remota. Se va a usar el driver ALSA.

7.2.1. Escenario 2: Esquema de la red

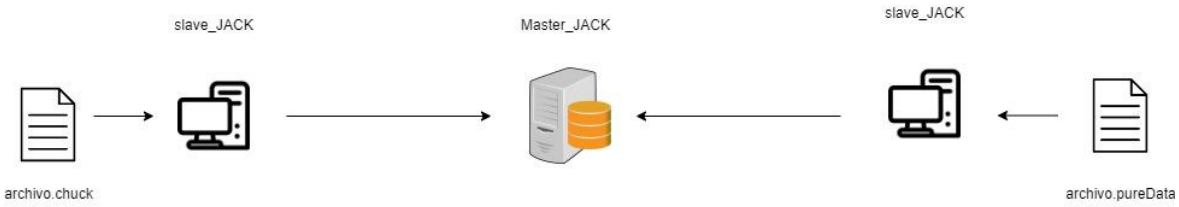


Ilustración 7 Esquema escenario_2

Este es el esquema que se va a seguir para realizar el segundo escenario, en donde tenemos un ordenador central que es donde se va a ejecutar el servidor JACK master y que va a reproducir el sonido, que le lleguen de los distintos esclavos. Luego tenemos dos ordenadores que cada uno ejecutar un servidor JACK slave, cada uno hará uso de uno de los clientes propuesto en el trabajo de la asignatura.

Todos los equipos, tanto el master como los equipos slaves van a seguir la siguiente estructura de protocolos:

PROTOCOLOS
MDNS
UDP
IGMP
Eth

Tabla 8 Torre de protocolos escenario_2

Estos protocolos están explicados en el apartado 2.

7.2.2. Escenario 2: Configuración del servidor

Para poder configurar el servidor habrá que realizar la instalación propuesta el apartado 5.2.1.1. Entonces iniciaremos un servidor JACK en el ordenador maestro y tras esto hay que ejecutar en el comando `jack_load netmanager`, esto es para que el servidor JACK sea un servidor maestro. En los ordenadores esclavos, en vez de iniciar el servidor JACK, usando el comando `jackd -d alsa`, hay que usar el comando `jackd -d net -a ip_multicast -p puerto`, para poder conectarnos con el servido JACK Maestro.

7.2.3. Escenario 2: Tests y Pruebas del Escenario

Los comandos que se van a utilizar son los siguientes:

1. Ordenador Maestro:

    a. `qjackctl`

    b. `jack_load netmanager`

2. Ordenador Esclavo-Pure Data

    a. `jackd -d net -a ip_multicast -p port`

    b. `qjackctl`

    c. `./pd -jack`

3. Ordenador Esclavo-Chuck

    a. `jackd -d net -a ip_multicast -p port`

    b. `qjackctl`

    c. `./chuck --driver:jack /dir/test_prueba.ck`

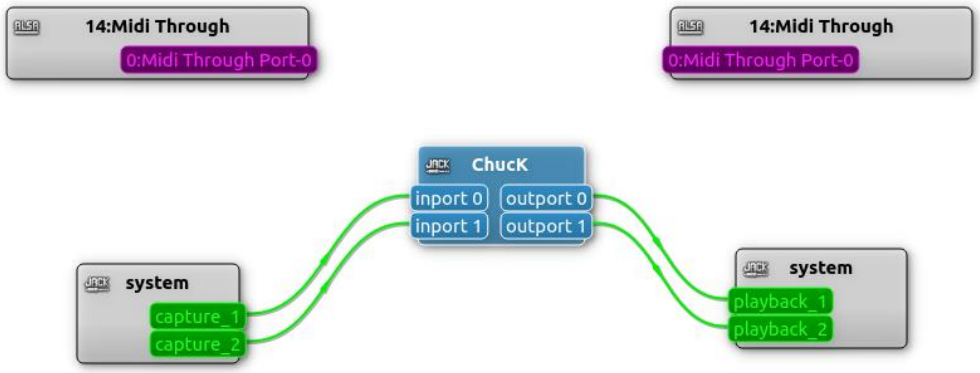


Ilustración 8 Esclavo_1-Chuck

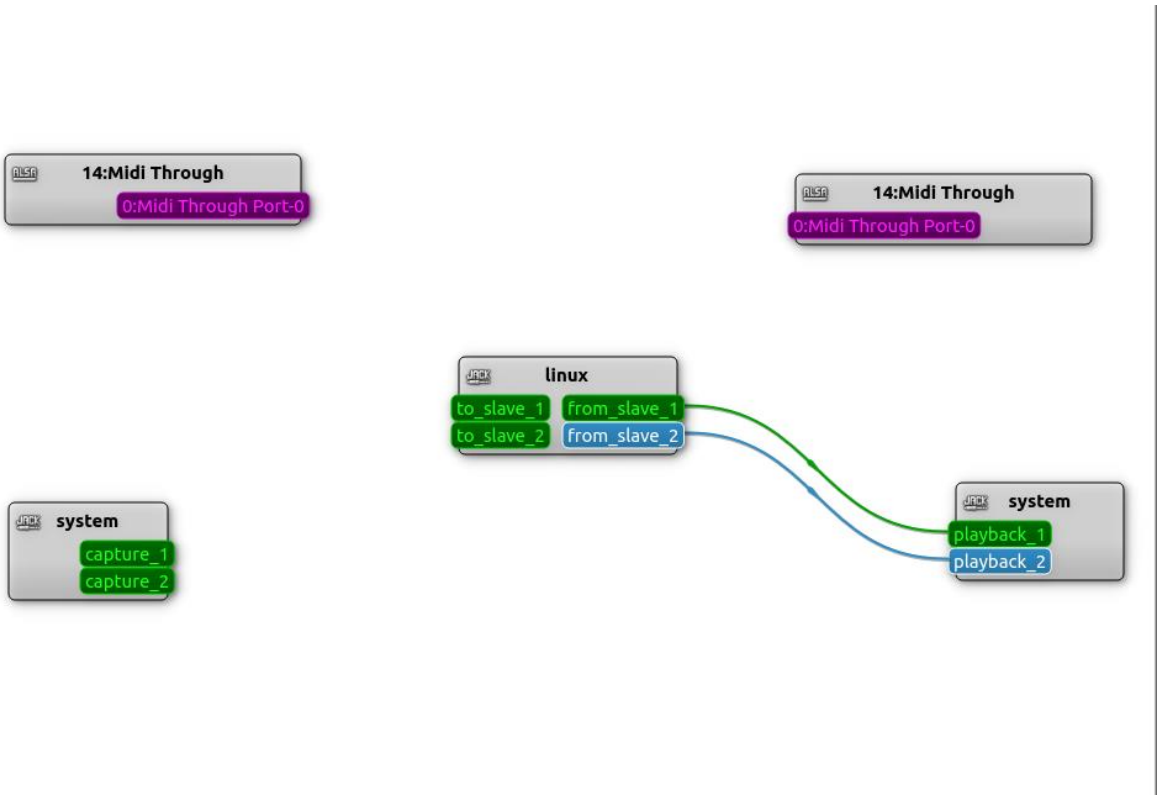


Ilustración 9 Maestro

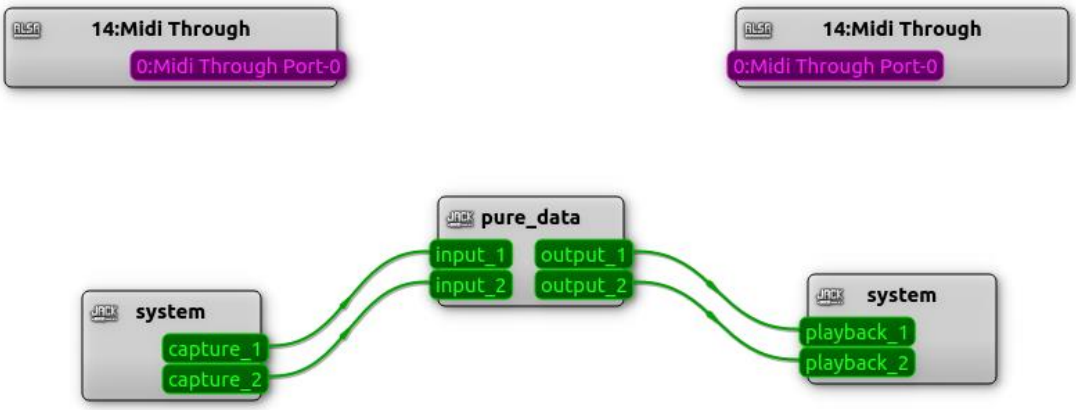


Ilustración 10 Esclavo-Pure Data

En la ilustración 7, no salen los dos esclavos, porque no puede tomar fotos en su día en el laboratorio y mi ordenador no soportaba tres clonaciones de máquinas virtuales.

7.2.4. Escenario 2: Análisis del intercambio real de mensajes de red

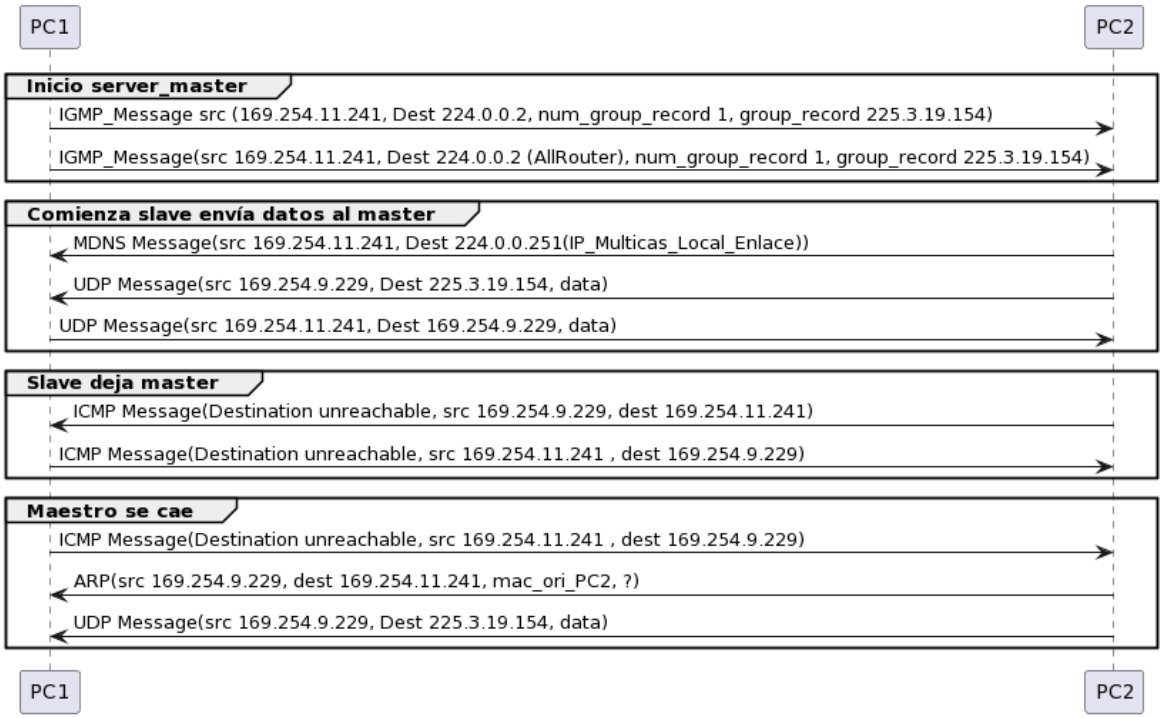


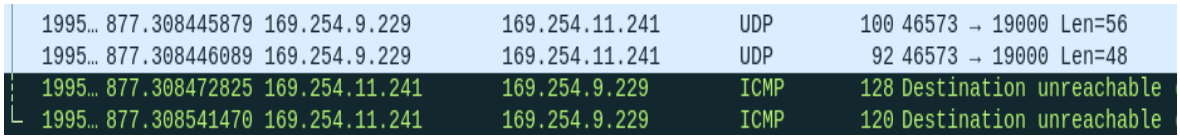
Ilustración 11 Paso de diagrama de mensajes del escenario_2

Este es el diagrama de paso de mensajes entre un maestro (PC1) y un esclavo (PC2), sería exactamente igual si hubiese un tercero, ya que actualmente JACK2 no permite la comunicación entre esclavos.

A continuación, voy a poner capturas de pantalla de la información obtenida por Wireshark, el nombre de la ilustración refleja a que parte del diagrama de mensajes indica:

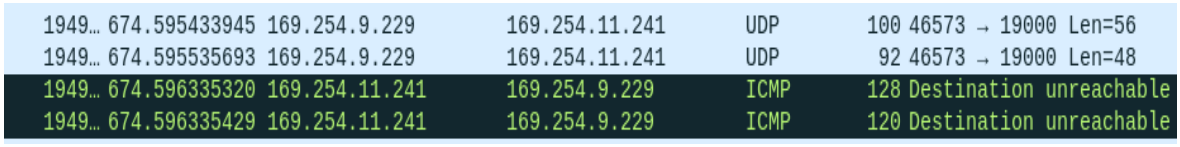
683	41.401096377	169.254.9.229	224.0.0.22	IGMPv3	56	Membership Report / Jo
684	41.475860840	169.254.9.229	224.0.0.22	IGMPv3	56	Membership Report / Jo
1195	71.532005604	fe80::a00:27ff:fe4c…	ff02::fb	MDNS	109	Standard query 0x0000
1196	71.532005839	169.254.11.241	224.0.0.251	MDNS	89	Standard query 0x0000
1213	71.978180176	169.254.11.241	225.3.19.154	UDP	688	19000 → 19000 Len=644
1216	71.981027808	169.254.9.229	169.254.11.241	UDP	688	46573 → 19000 Len=644

Ilustración 12 Inicio_server_master y comienza slave envía datos al master



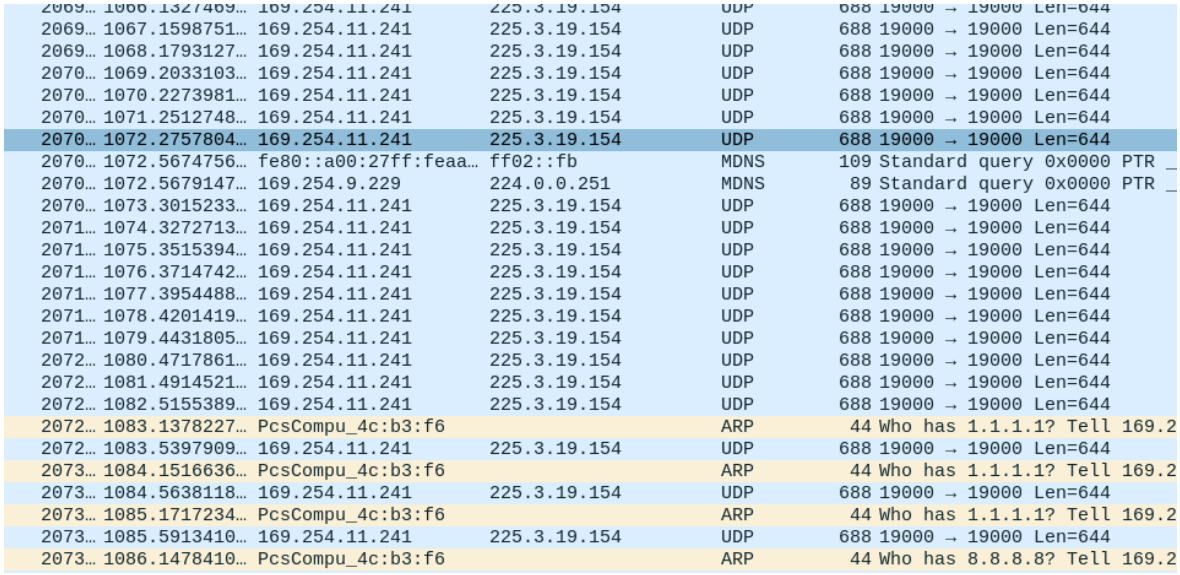
1995…	877.308445879	169.254.9.229	169.254.11.241	UDP	100 46573 → 19000 Len=56
1995…	877.308446089	169.254.9.229	169.254.11.241	UDP	92 46573 → 19000 Len=48
1995…	877.308472825	169.254.11.241	169.254.9.229	ICMP	128 Destination unreachable
1995…	877.308541470	169.254.11.241	169.254.9.229	ICMP	120 Destination unreachable

Ilustración 13 Slave deja a master. Captura en el ordenador esclavo.



1949…	674.595433945	169.254.9.229	169.254.11.241	UDP	100 46573 → 19000 Len=56
1949…	674.595535693	169.254.9.229	169.254.11.241	UDP	92 46573 → 19000 Len=48
1949…	674.596335320	169.254.11.241	169.254.9.229	ICMP	128 Destination unreachable
1949…	674.596335429	169.254.11.241	169.254.9.229	ICMP	120 Destination unreachable

Ilustración 14 Slave deja a master. Captura en el ordenador maestro.



2069…	1067.1598751…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2069…	1068.1793127…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2070…	1069.2033103…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2070…	1070.2273981…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2070…	1071.2512748…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2070…	1072.2757804…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2070…	1072.5674756…	fe80::a00:27ff:feaa…	ff02::fb	MDNS	109 Standard query 0x0000 PTR _
2070…	1072.5679147…	169.254.9.229	224.0.0.251	MDNS	89 Standard query 0x0000 PTR _
2070…	1073.3015233…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2071…	1074.3272713…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2071…	1075.3515394…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2071…	1076.3714742…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2071…	1077.3954488…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2071…	1078.4201419…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2071…	1079.4431805…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2072…	1080.4717861…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2072…	1081.4914521…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2072…	1082.5155389…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2072…	1083.1378227…	PcsCompu_4c:b3:f6		ARP	44 Who has 1.1.1.1? Tell 169.2
2072…	1083.5397909…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2073…	1084.1516636…	PcsCompu_4c:b3:f6		ARP	44 Who has 1.1.1.1? Tell 169.2
2073…	1084.5638118…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2073…	1085.1717234…	PcsCompu_4c:b3:f6		ARP	44 Who has 1.1.1.1? Tell 169.2
2073…	1085.5913410…	169.254.11.241	225.3.19.154	UDP	688 19000 → 19000 Len=644
2073…	1086.1478410…	PcsCompu_4c:b3:f6		ARP	44 Who has 8.8.8.8? Tell 169.2

Ilustración 15Master se cae. Captura desde el esclavo

2012…	838.022256393	169.254.9.229	224.0.0.22	IGMPv3
2013…	838.836114407	169.254.9.229	224.0.0.22	IGMPv3
2018…	869.853849210	fe80::a00:27ff:feaa…	ff02::fb	MDNS
2018…	869.854260385	169.254.9.229	224.0.0.251	MDNS
2018…	869.854713323	127.0.0.1	224.0.0.251	MDNS
2018…	872.238354570	192.168.214.1	224.0.0.251	MDNS
2018…	872.550945000	172.16.153.1	224.0.0.251	MDNS
2018…	873.468110756	fe80::250:56ff:fec0…	ff02::fb	MDNS
2018…	873.740780316	fe80::250:56ff:fec0…	ff02::fb	MDNS
2026…	920.917687288	169.254.9.229	169.254.9.229	ICMP
2027…	923.927985416	169.254.9.229	169.254.9.229	ICMP
2027…	926.932553608	169.254.9.229	169.254.9.229	ICMP
2048…	1056.1480660…	169.254.9.229	169.254.9.229	ICMP
2049…	1059.1563723…	169.254.9.229	169.254.9.229	ICMP
2049…	1062.1646784…	169.254.9.229	169.254.9.229	ICMP

Ilustración 16 Master se cae. Captura desde el maestro

8. Interfaz gráfica de administración del servidor

La interfaz gráfica que nos permite administrar el servidor JACK es qjackctl, es una herramienta con la cual, podemos arrancar y parar el servidor, además nos proporciona logs por si existiera algún problema al iniciar el servidor o por si algún esclavo no haya podido conectarse. También esta herramienta nos ofrece la posibilidad de cambiar la configuración del servidor ya sea por ejemplo el controlador con el que queramos inicializar el servidor o la tasa de muestro. Además, nos permite también controlar si queremos que llegue transporte al servidor o no y una pantalla donde se visualiza esta situación.

Por último, lo más importante de esta interfaz gráfica es que nos permite visualizar un mapa de conexiones con las distintas aplicaciones de audio que se están utilizando en el sistema, además de los esclavos que están conectados a nuestro servidor.

9. Deficiencias del servicio

Este servicio está muy bien conseguido en cuanto información a lo hora de instalar y configurar el servidor, pero no hay mucha información sobre se usó a nivel de red.

También, otra deficiencia es que los esclavos solo puedan conectarse con el maestro, y no entre ellos ya que por ejemplo si tú tienes que pasarle algún archivo de audio algún

compañero, pero no quieres saturar al maestro ya que todavía no es definitivo, no se puede hacer ya que la única manera comunicarse es esclavo con maestro.

Por último, este servicio no nos permite manejar nada en tema de seguridad al arrancar el servicio, ya sea haciendo uso del servidor de manera local como remota, no se si debe a que no es información importante de proteger o que como se va a trabajar en redes locales en donde la persona conoce el estado de red y su configuración. Lo único que he visto es que, si denegamos en un ordenador el acceso a ICMP, no podrá existir comunicación entre esclavos y maestros.

10. Ampliaciones/mejoras del servicio

Realizaría un tutorial más extendido sobre los usos de la herramienta gráfica qjackctl, ya que l principio es poco intuitiva y también mejoraría que se pudiese establecer también comunicación entre esclavos y no solo con el maestro.

11. Incidencias y principales problemas detectados

El principal problema que he encontrado ha sido que no haya tutoriales sobre el uso de los sintetizadores usados con JACK2, he tenido que mirar varias veces la información que nos proporciona los clientes con el comando –help para poder establecer comunicación con el servidor JACK. Después también el uso a nivel de red de JACK2 aunque hay un tutorial por parte de los desarrolladores de JACK2 en la página de la referencia [3], no existen ejemplos en internet sobre su uso, solo esta página.

12. Resumen y Conclusiones

En conclusión, me ha parecido un trabajo muy interesante, ya que me he tenido que enfrentar a la instalación de un servicio desde código fuentes, que nunca lo había hecho, también he podido aprender más sobre el tema de manejo de sonidos de un ordenador y de poder ver diversas maneras de producir música electrónica.

Por último, aunque no haya podido observar con este servicio algunas de las herramientas dadas en las práctica, si que he notado que he entendido que estaba haciendo como por al generar el archivo de arranque en el sistema , o cuando he tenido que modificar el kernel y por qué había que tocar este archivo o por qué había que reiniciar el ordenador.

Por todo esto siento que ha sido un trabajo satisfactorio, ya que, aunque no haya podido profundizar he podido introducirme un poco en el tema de la música electrónica, también he probado un estilo de instalación distinta a la que estaba acostumbrado y lo más importante he notado que los conocimientos adquiridos en las prácticas me han servido para poder comprender que hacía en cada momento.

ANEXO A: Parámetros de configuración y comandos de gestión del servidor

No procede

ANEXO B: Parámetros de configuración y comandos de gestión del cliente

No procede