ESCUELA DE INGENIERÍA EN COMPUTACIÓN

Sistemas Operativos

PROYECTO DE STREAMING

Profesor:
Esteban Arias

Equipo:
Esteban González Damazio 2014098850
Markus 69
Jose Pablo Vargas Campos 2013116365

I semestre 2018

Martes 5 de junio
San José, Costa Rica

Diseño

El sistema consta de dos partes, Servidor y Clientes. El servidor se desarrolla en C y vive dentro de una máquina remota host que corre el sistema operativo Debian y se encuentra en los servidores de DigitalOcean. El servidor es headless. Se aprovechan los protocolos RTSP/RTP (Real Time Streaming Protocol/Real-time Transport Protocol) para la comunicación de datos sobre la red. El protocolo RTSP permite la transmisión de datos para realizar streaming sin necesitar enviar todo el archivo. RTSP también se encarga de controlar el control de playback del stream entre los endpoints con instrucciones para pausar, reanudar o elegir un nuevo punto para el playback del archivo. Además se utilizan sockets para controlar la comunicación de datos que no son de streaming entre el Servidor y los Clientes, estos sockets se han utilizado con la ayuda de la librería opensource de ZeroMQ que tiene altos niveles de calidad y robusticidad. El cliente por otra parte se ha desarrollado en java, con su propia biblioteca en java de ZeroMQ para comunicarse con el servidor y con un sink de la biblioteca de Jvlc para reproducir videos, la cual está basada en el reproductor open source VLC.

Bibliotecas

Gstreamer

https://gstreamer.freedesktop.org/documentation/index.html

Biblioteca en C que se utiliza para realizar las operaciones de servidor que se encargan de codificar los datos de video MP4, empaquetarlos a RTP y controlar las llamadas de control de playback RTSP. También se encarga de hacer lo mismo para los datos de audio Vorbis Ogg, sin embargo las limitaciones de la biblioteca cliente Jvlc no permiten utilizar los datos de audio Vorbis Ogg (.ogg).

Jvlc

https://github.com/caprica/vlcj

Biblioteca en Java que permite implementar VLC dentro de la vista de Swing. Se utiliza como el sink para poder desempacar los paquetes RTP, decodificar los datos de video

en formato MP4 y controlar las llamadas RTSP. VLC tiene limitaciones para procesar los datos de audio Vorbis Ogg por lo que no puede utilizarse para la implementación de playback de música.

ZeroMQ

http://zeromq.org

ZeroMQ es una biblioteca de mensajería asíncrona de alto rendimiento, diseñada para su uso en aplicaciones distribuidas o concurrentes. Proporciona una cola de mensajes, pero a diferencia del middleware orientado a mensajes, el sistema ZeroMQ puede ejecutarse sin un intermediario de mensajes dedicado.

Servidor

Se compone de dos partes. La primera parte utiliza la biblioteca Gstreamer para realizar la comunicación de datos por TCP/IP utilizando el protocolo RTP/RTSP como una capa de abstracción sobre TCP/IP. El servidor es thread-safe y puede atender a múltiples clientes al mismo tiempo sin problemas de concurrencia aun cuando diferentes clientes solicitan datos del mismo archivo. La segunda parte utiliza la biblioteca Zeromq para realizar el intercambio de mensajes con cada cliente por individual sin problema de confusión de clientes ni confusión de mensajes.

gstserver**urlgeneration.h**: utilizado para generar los URL de los streams, al tener su propio archivo asegura consistencia entre los varios programas que lo usan.

gstserver**zhelpers.h**: funciones útiles para ser reutilizadas constantemente junto a la biblioteca zeromq.

gstserver**mp4-streaming.c (-o mp4)**: La parte del servidor que hace el streaming. Programa que se encarga de crear los streams y asignar URLs. Utiliza `find` junto con pipes para crear los streams dinámicamente. Utiliza la biblioteca gstreamer para realizar el streaming del payload por medio del protocolo RTSP hacia el cliente por medio de un URL creado dinámicamente para cada archivo.

gstserver**unused.c**: Macros para evitar warning de gcc sobre parametros de funcion no utilizados.

gstserver**server.c (-o server)**: El servidor encargado de las conexiones de los clientes, realiza tareas como comunicar la lista de canciones, y en general la transferencia de mensajes entre el servidor y el usuario por medio de zeromq.

gstserver**makefile**: make file :p

gstserver**run.sh**: para correr el servidor, contiene los siguientes comandos:

```
#!/bin/
bash
```

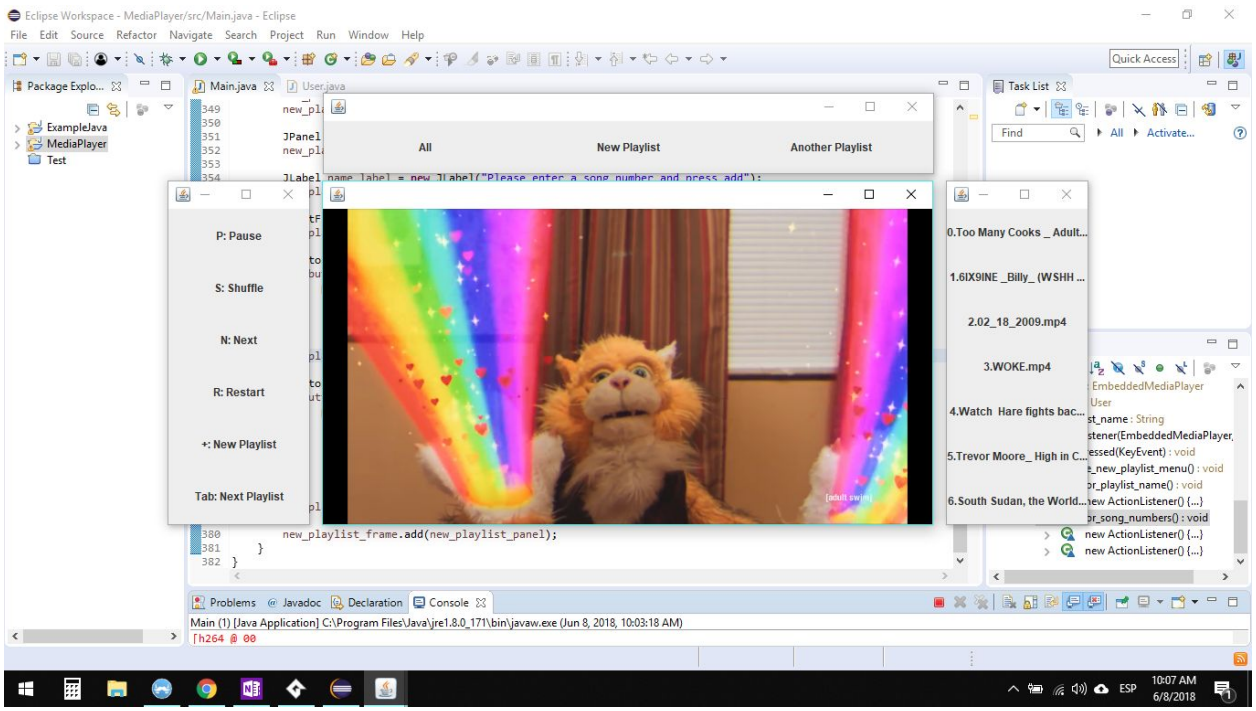
```
trap "trap - SIGTERM && kill -- -$$" SIGINT SIGTERM EXIT


find video -type f -print0 | xargs -0 ./mp4 > stream.log & ./server >
server.log & tail -f server.log
```

Cliente

MediaPlayer\src**Main.java** & MediaPlayer\src**User.java**

Desarrollado en Java con la biblioteca Jvlc. Se comunica con el servidor para recibir la lista de links para streaming de los datos, además puede enunciar su usuario y enunciar cuales vídeos quiere streamer para el log del servidor.

A continuación se muestra una imagen de como se ve el cliente activo:



En el panel izquierdo se ven los controles de la aplicación, el panel superior contiene los playlists creados por el usuario, y el panel derecho las canciones del playlist actual.

Pruebas Realizadas

Comunicación de mensajes: Se ha probado el servidor con 5 clientes concurrentes en un loop donde cada uno envía un mensaje diferente y como petición cada uno debe de recibir su respuesta apropiada del servidor. De esta manera se ha confirmado que el servidor puede manejar una demanda de mensajes concurrentes sin confundir la respuesta a ninguno de los mensajes. Entre los clientes se han utilizado clientes simplificados de C y de Java al mismo tiempo para realizar esta prueba.

gstserver**client**: cliente ejemplo en c utilizado para pedir mensajes

ExampleJava\src**main.java**: programa ejemplo en java para pedir mensajes

Visualización de stream RTSP:

Se ha probado la visualización de los streams RTSP desde el cliente de Java y desde el reproductor abierto VLC. Se ha probado la utilización de estos estos streams viendo el mismo archivo desde varios clientes diferentes para confirmar la habilidad de hacer streaming concurrente sin problemas entre varios clientes que requieren diferentes mensajes.

Estado del proyecto

Toda su funcionalidad está completa, el streaming se realiza con archivos de vídeo en vez de archivos de audio como extra.

Posibles mejoras

Recibir el stream de música por medio de esta biblioteca de RTSP/RTP para Java
http://www.smaxe.com/product.jsf?id=juv-rtsp-client
Junto a esta biblioteca que proporciona codecs para para Java del formato de audio Ogg Vorbis
http://www.jcraft.com/jorbis/
El servidor ya tiene la habilidad para originar el stream RTSP/RTP de audio necesario.

Bibliografia

Real Time Streaming Protocol (RTSP)
https://tools.ietf.org/html/rfc2326

Servidor de RTSP dentro de Gstreamer.

https://github.com/thaytan/gst-rtsp-server/tree/master/docs

Libreria Gstreamer para generar pipelines de archivos multimedia.
https://gstreamer.freedesktop.org/documentation/

Popen() para correr comandos y parsear su output. Crea un pipe, hace un fork e invoca al shell.
http://man7.org/linux/man-pages/man3/popen.3.html