Informe de Proyecto

CURSO 2016-2017

Automatización del proceso de creación de un team en el protocolo P2PSP

JOSE CARLOS FUENTES ANGULO
NICOLÁS RIAL CAPILLA
VICTOR EUGENIO RODRÍGUEZ RUÍZ

TECNOLOGÍAS MULTIMEDIA

4º GRADO EN INGENIERÍA INFORMÁTICA

Universidad de Almería

Trabajo de Tecnologías Multimedia: P2PSP

El P2PSP (Peer to Peer Straightforward Protocol) es un protocolo de comunicación de la capa de aplicación para el streaming de contenido multimedia sobre Internet diseñado por investigadores de la Universidad de Almería. Este protocolo se basa en dos ideas básicas:

1. A pequeña escala, todos los peers se comunican entre sí, todos con todos, utilizando mensajes punto a punto (unicast).
2. Los peers deben ser tan solidarios con el resto de peers como estos lo son con él, o de lo contrario, no podrán formar parte de la red P2P. Dicha solidaridad se expresa en términos de ancho de banda, por tanto, esto significa que, para que un peer forme parte de la red, debe de enviar a esta, en promedio durante cada cierto intervalo de tiempo, tanto como recibe de ella.

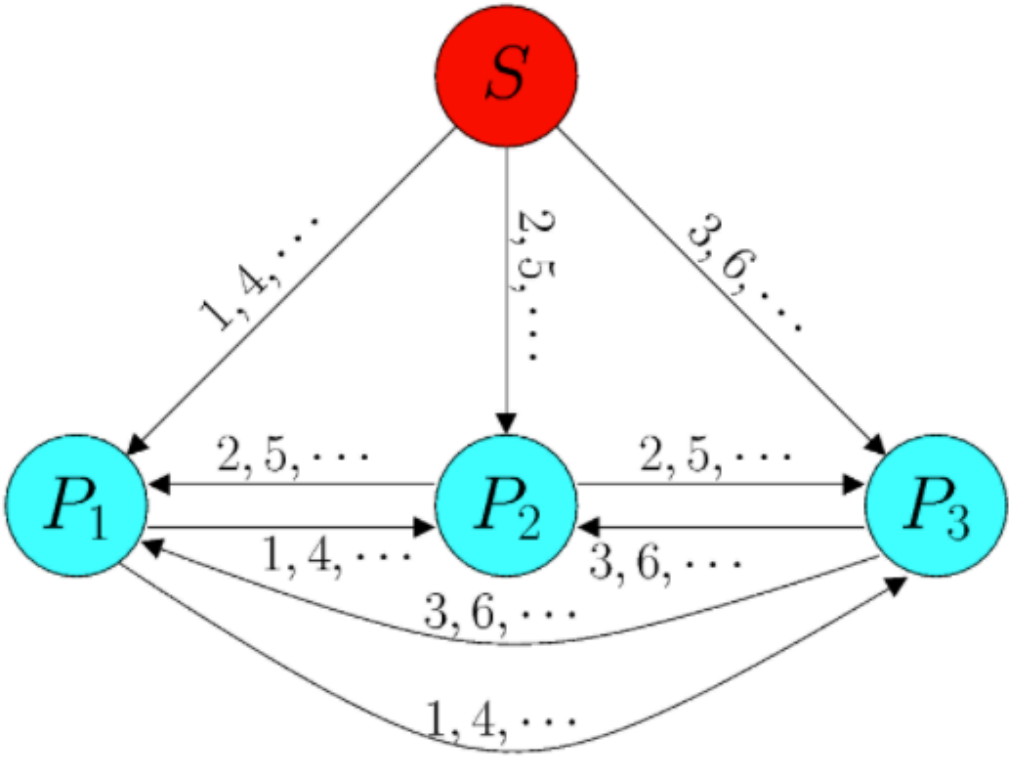
El P2PSP ha sido especialmente definido para realizar streaming de secuencias en directo, es decir, puede ser utilizado para transmitir a través de una red de comunicaciones basada en la conmutación de paquetes (como lo es la red Internet) un evento que se está produciendo en ese momento, aunque también puede ser aplicado en la difusión de secuencias ya almacenadas.

Estas son algunas de las características del protocolo P2PSP:

- El protocolo está construido de forma modular. El número de módulos que se usan depende del tipo de sistema que vayamos a implementar.
- Al ser modular es un protocolo que responde a cambios y errores, por lo que, si se produce un error, se convierte en un fallo aislado y no produce un efecto cascada
- La configuración por defecto del sistema para implementar el protocolo tiene un coste computacional muy bajo, el resto de módulos son complementos que añaden funcionalidades extras (conexión a través de NATs, streaming en paralelo, etc)
- El sistema P2PSP ha sido creado para servicios de streaming en tiempo real, pero puede usarse también para desplegar sistemas híbridos C/S-P2PSP
- Los peers se pueden alojar en redes privadas incluso si están tras un NAT.

Cuando halamos de P2PSP, la información se transmite de dos formas completamente distintas.

- Como un flujo o stream, como una secuencia sin fin de datos que transmite cierta información, la cuál siempre se transmite mediante TCP
- Como una colección de chunks, siendo un chunk un trozo del stream. Todos los chunks tienen el mismo tamaño, un chunk pequeño reduce la latencia, pero incrementa la cantidad de información que se procesa. Todos los chunks son transmitidos mediante UDP



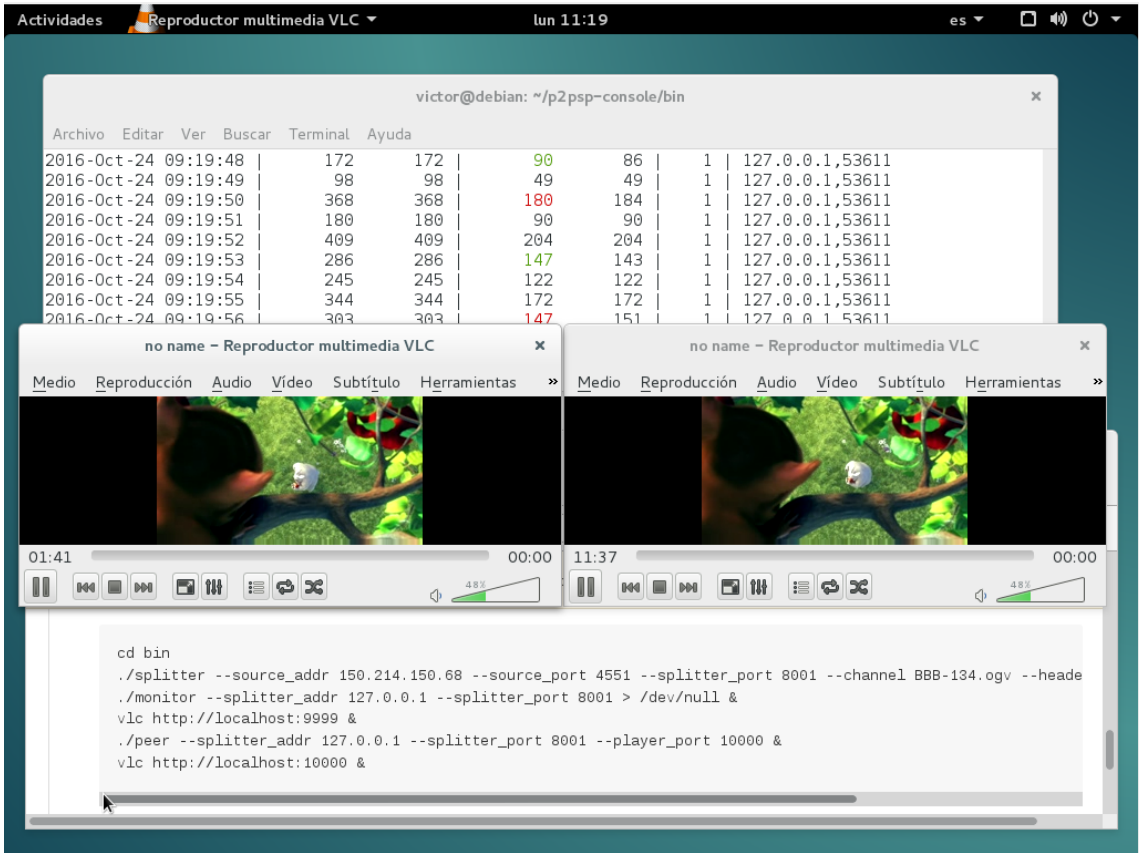
En esta imagen podemos observar un P2PSP team. Las flechas y sus etiquetas indican la transmisión de chunks. S envía un chunk diferente a cada peer los cuales han sido escogidos mediante el modelo Round-Robin, los peers se envían entre ellos los chunks que les faltan para completar el stream

Entidades principales:

Un sistema P2PSP se compone de las entidades que se describen a continuación:

- **Source(O):** Es el que crea el stream que será transmitido posteriormente en la red P2PSP. Normalmente es un servidor de streaming que usa un protocolo http como Icecast. El source controla el bit-rate dentro de la red P2PSP.
- **Player(L):** Decodifican y reproducen el stream. Generalmente, usan técnica de interpolación de señales para cubrir los huecos cuando faltan partes en el stream. Esto es especialmente útil porque la pérdida de chunks tiende a incrementarse con el tiempo en una red P2PSP.
- **Splitter(S):** Es el que recibe el stream del source y lo divide en chunks del mismo tamaño, tras esto envía los chunks a los peers.
- **Peer(P):** Recibe los chunks del splitter y de otros peers e igualmente envía chunks a otros peers, reensambla el stream uniendo los chunks y se lo envía al player. Advertir que el player y el peer suelen estar alojados en la misma máquina
- **Team:** Es la composición de un splitter, un player (un peer en consecuencia) y al menos otro peer.

Una vez explicados los conceptos básicos para entender nuestro proyecto procedemos a explicarlo. Nuestra tarea en este proyecto ha sido la de programar un pequeño script que automatice la creación de un team con un número de peer a petición del usuario.



En la imagen anterior, se ha empleado el código mostrado en dicha imagen para ejecutar un team local, que además se aprecia que funciona correctamente. Dicho código está ubicado en la página web de GitHub > P2PSP > I want to use it > Run a local team.

Éste es el código de nuestro proyecto:

```
#!/bin/bash
rm /home/hackerman/TM/*.txt
cd /home/hackerman/p2psp-console/bin/
```

Acceder a la carpeta que contiene los archivos a usar: monitor, peer y splitter.

```
while getopts "a:" opt; do
  case ${opt} in
    a)
      NUM_PEERS="${OPTARG}"
      ;;
    \?)
      echo "Invalid option: -${OPTARG}"
      ;;
  esac
done
```

El programa preguntará al usuario cuántos peers quiere ejecutar. Debe introducirse un número entero. Si se introducen caracteres, el programa rechazará esa entrada. La entrada introducida se guardará en una variable para su posterior uso.

```
TIME_RANDOM=`shuf -i 10-600 -n 1`
TIME_NETCAT=`shuf -i 10-600 -n 1`
```

En estas variables se guardarán un tiempo al azar.

```
timeout $TIME_RANDOM ./splitter --source_addr 150.214.150.68 --source_port 4551
--splitter_port 8001 --channel BBB-134.ogv --header_size 30000 >
/home/hackerman/TM/splitter.txt &
```

Ejecutar el splitter en el tiempo aleatorio generado anteriormente.

```
timeout $TIME_RANDOM ./monitor --splitter_addr 127.0.0.1 --splitter_port 8001 >
/home/hackerman/TM/monitor.txt &
```

Ejecutar el monitor en el tiempo aleatorio generado anteriormente.

```
vlc http://localhost:9999 &
```

Ejecutamos VLC para visualizar el stream.

```
for ((i=0; i < $NUM_PEERS; i++))
```

Ejecutar el bucle For tantas veces como se haya indicado previamente en la variable ‘número de peers’.

```
do
PORT_RANDOM=`shuf -i 2000-65000 -n 1`
echo $PORT_RANDOM
```

Generar y mostrar por pantalla un puerto aleatorio.

```
./peer --splitter_addr 127.0.0.1 --splitter_port 8001 --player_port $PORT_RANDOM >
/home/hackerman/TM/peers$PORT_RANDOM.txt &
sleep 0.5
```

Crea tantos peers como el usuario haya indicado, cada peer debe crearse en un puerto diferente. Envía la información del peer al txt en la ruta indicada. Esperamos 0,5” para la creación del siguiente

```
timeout $TIME_NETCAT nc localhost $PORT_RANDOM > /dev/null &
done
```

Netcat se conectará al puerto generado anteriormente al azar, con un tiempo de vida aleatorio