

Redes de Computadoras

Obligatorio 2 - 2018

Facultad de Ingeniería
Instituto de Computación
Departamento de Arquitectura de Sistemas

Nota previa - IMPORTANTE

Se debe cumplir íntegramente el "Reglamento del Instituto de Computación ante Instancias de No Individualidad en los Laboratorios", disponible en el EVA.

En particular está prohibido utilizar documentación de otros estudiantes, de otros años, de cualquier índole, o hacer público código a través de cualquier medio (EVA, news, correo, papeles sobre la mesa, etc.).

Introducción

Forma de entrega

Una clara, concisa y descriptiva documentación es clave para comprender el trabajo realizado. La entrega de la tarea consiste en un único archivo `obligatorio2GrupoGG.tar.gz` que deberá contener los siguientes archivos:

- Un documento llamado `Obligatorio2GrupoGG.pdf` donde se documente todo lo solicitado en la tarea. GG es el número del grupo. La documentación deberá describir claramente su solución, las decisiones tomadas, los problemas encontrados y posibles mejoras
- Los programas solicitados con nombre **servidor.cc** y **cliente.xx** (xx es la extensión correspondiente al lenguaje utilizado).
- Un directorio `extras` incluyendo cualquier otro archivo que considere relevante.

La entrega se realizará en el sitio del curso, en la plataforma EVA.

Fecha de entrega

Los trabajos deberán ser entregados antes del 23/09/2018 a las 23:30 horas. No se aceptará ningún trabajo pasada la citada fecha y hora. En particular, no se aceptarán trabajos enviados por e-mail a los docentes del curso.

Objetivo del Trabajo

Aplicar los conceptos teóricos de capas de aplicación y transporte y la utilización del API de sockets TCP y UDP.

Descripción general del problema

Se desea implementar una aplicación de transmisión de video *on-demand* que permita el intercambio de información entre múltiples usuarios.

Problema a resolver

Se desea implementar un servicio de transmisión de video en línea y a demanda para usuarios en Internet. El mismo utilizará los protocolos TCP para el canal de control y UDP para el canal de datos. Contará con un servidor que recibe los pedidos de los clientes y transmite a cada cliente el video solicitado.

Además, se desea contar con una aplicación cliente para reproducir los videos recibidos.

Descripción del servidor

El servidor es la aplicación que permite hacer la transmisión de video. Este servicio tiene la capacidad de realizar la transmisión utilizando el protocolo UDP.

Por cada cliente que lo solicite, se realiza la transmisión leyendo los datos a enviar desde un archivo de video alojado localmente.

Se asumirá que el servidor contiene solo un archivo de video para transmitir.

El servidor acepta conexiones TCP en una dirección IP y puerto conocidos para el control de las transmisiones.

El control brinda las opciones de: iniciar una transmisión, pausar y reiniciar una transmisión y finalizar una transmisión.

Descripción del cliente

El cliente realizará la reproducción del video recibido.

Para esto brindará al usuario un menú con las opciones de REPRODUCIR, PAUSAR y FINALIZAR.

Al seleccionar la opción de REPRODUCIR el cliente se conecta al servidor (si no está conectado aún), realiza el pedido del video y queda leyendo el *stream*, reproduciendo el video en pantalla.

Con la opción de PAUSAR, el cliente debe informar al servidor que desea suspender la reproducción temporalmente. Si luego de una pausa se selecciona REPRODUCIR nuevamente, el video continúa a partir del instante en el que se suspendió.

La opción FINALIZAR permite avisar el servidor que se desea terminar la recepción del video en ese momento. Si luego de una finalización se selecciona REPRODUCIR nuevamente, el video comenzará desde el principio.

Se pide

- a) Diseñe el protocolo y los mensajes necesarios para implementar los canales de datos y de control descritos.

- b) Implemente el servidor y el cliente.

El servidor debe ser implementado en C/C++. No hay restricciones en el lenguaje del cliente.

Para la transmisión del video puede asumir que el tamaño de un *frame* de video es siempre menor que el tamaño máximo de un datagrama UDP.

- c) Explique detalladamente como debería cambiar su solución si el supuesto anterior ("el tamaño de un *frame* de video es siempre menor que el tamaño máximo de un datagrama UDP") deja de ser válido. Su propuesta debe ser robusta a la pérdida o reordenamiento de paquetes. Justifique.

Para la reproducción de video se utilizará la biblioteca OpenCV [1]. Esta biblioteca esta disponible para C/C++, Java y Python. En el Anexo se describen las funciones mínimas necesarias de OpenCV que deberá utilizar (para el lenguaje C++).

Se recomienda también revisar y utilizar como base el código de ejemplo de uso de sockets dado en el curso.

Anexo - Ejemplo de uso de OpenCV

A continuación se muestran ejemplos de uso de la API en C++ de OpenCV. Puede utilizar esto como base para su desarrollo pero es recomendable que estudie la documentación de OpenCV [2].

Código de inicialización en servidor:

```
#include "opencv2/opencv.hpp"

Mat frame;           //array que representa los frames del video
vector<uchar> encoded; //vector para almacenar el frame codificado en jpeg
VideoCapture cap(video.mp4); //Obtener el video desde archivo.

//Crea una nueva ventana con nombre servidor
namedWindow("servidor", CV_WINDOW_AUTOSIZE);
```

Obtener un frame del archivo y mostrarlo en pantalla:

```
cap >> frame;           //obtener nuevo frame
imshow("servidor", frame); //muestra la imagen en la ventana servidor
WaitKey(1000/30);       //intervalo entre frames en ms
```

Codificar la imagen en jpg para ser transmitida:

```
vector<int> compression_params;
compression_params.push_back(CV_IMWRITE_JPEG_QUALITY);
compression_params.push_back(80);

imencode(".jpg", frame, encoded, compression_params);
//la imagen codificada queda en encoded
```

Decodificar y mostrar la imagen:

```
namedWindow("cliente", CV_WINDOW_AUTOSIZE);

//matriz con la imagen recibida.
//size es el tamaño de los datos recibidos
//en buffer se encuentran los datos recibidos
Mat rawData = Mat(1, size, CV_8UC1, buffer);
Mat frame = imdecode(rawData, CV_LOAD_IMAGE_COLOR); //decodifica la imagen

imshow("cliente", frame); //muestra la imagen en la ventana cliente
waitKey(1);
```

Si se utiliza C/C++ se recomienda compilar utilizando CMake. La siguiente es una configuración de ejemplo:

```
cmake_minimum_required(VERSION 2.8)
project( obligatorio2 )
find_package( OpenCV REQUIRED )
add_executable( servidor servidor.cc )
target_link_libraries( servidor ${OpenCV_LIBS} )
add_executable( cliente cliente.cc )
target_link_libraries( cliente ${OpenCV_LIBS} )
```

Luego, para compilar debe hacer:

```
cmake .  
make
```

Tenga en cuenta que seguramente será necesario instalar la biblioteca OpenCV y la herramienta cmake en su sistema operativo.

Por ejemplo, para instalar OpenCV en un SO tipo Debian debe hacer:

```
sudo apt-get install libopencv-dev
```

Se sugiere utilizar el SO Ubuntu o Lubuntu en sus últimas versiones. Si no cuenta con uno, puede crear una máquina virtual descargando la imagen de Lubuntu en [3].

Referencias y Bibliografía Recomendada

[1] Open Source Computer Vision Library. opencv.org

[2] Documentación de OpenCV. <http://docs.opencv.org/2.4/index.html>

[3] Imagen ISO de Lubuntu 17.04.

<http://cdimage.ubuntu.com/lubuntu/releases/17.04/release/lubuntu-17.04-desktop-i386.iso>