REDES DE COMUNICACIONES II

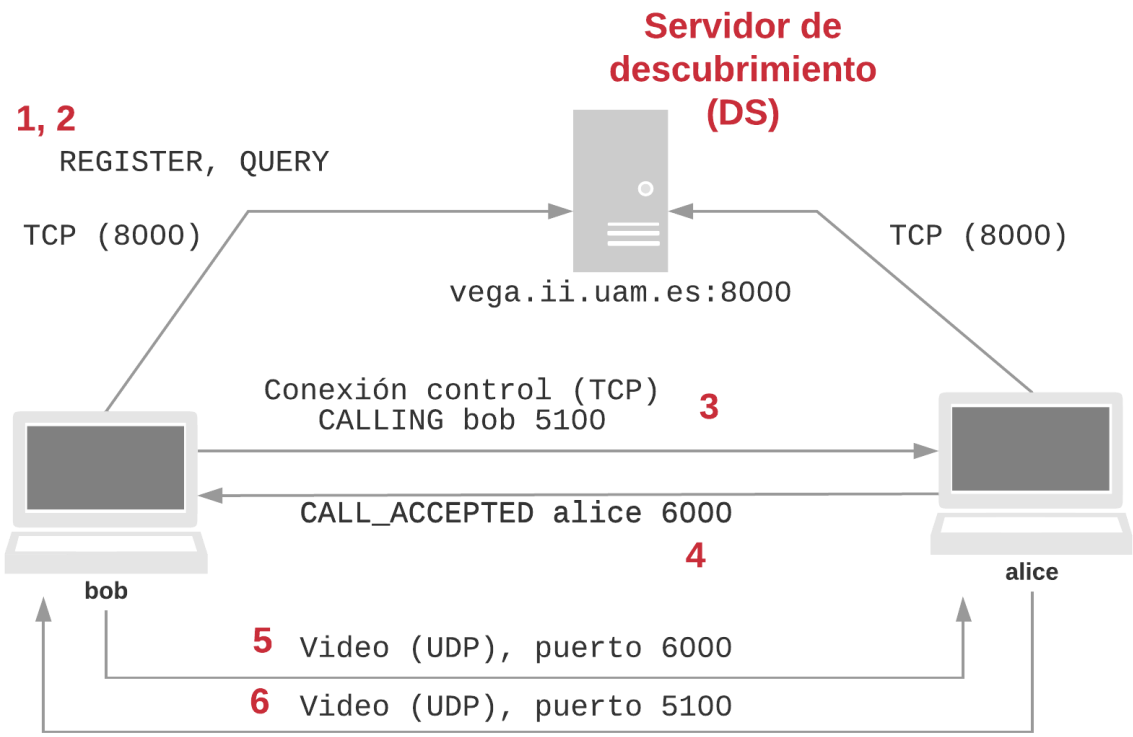
Práctica 3 - Multimedia

Volver a: Prácticas

3 Comunicaciones

Arquitectura

A grandes rasgos, la arquitectura del sistema es la que se muestra en la siguiente figura:



En ella se aprecia que hay dos tipos básicos de comunicaciones:

- Conexiones de control TCP con el DS y el nodo destino (alice)
- Datagramas UDP de datos entre los pares (video)

Para realizar una llamada, una secuencia tipo deberá ser similar a la siguiente (mensajes en rojo en la figura):

- **Mensaje 1**: La aplicación registra (o actualiza) el nick del usario en el DS cada vez que ésta se lanza, o el usuario solicitad un cambio de nick, respectivamente. Posteriormente, la aplicación deberá permitir al usuario iniciar una videollamada. Para ello:
  - El usuario podrá escribir directamente el nick deseado, si ya lo conoce.
  - La aplicación mostrará una lista de nicks registrados (obtenidos a través de LIST_USERS) y permitirá al usuario seleccionar uno.
- **Mensaje 2**: La aplicación pide al DS los datos necesarios del nick elegido: dirección IP, puerto y protocolos soportados. El protocolo a usar será el protocolo de máximo nivel común a los dos clientes.
- **Mensaje 3**: La aplicación inicia la conexión de control con el cliente, en la dirección IP y puerto especificados en el mensaje anterior.
- **Mensaje 4**: La otra parte aceptará o declinará la llamada con CALL_ACCEPTED o CALL_DENIED, respectivamente.
- **Mensajes 5 y 6**: Los datos de video se transmiten por UDP al puerto recogido en el comando CALL_ACCEPTED.

Obviamente, la aplicación debe ser capaz de hacer todo esto simultáneamente, atendiendo al interfaz gráfico, recibiendo y emitiendo video, y atendiendo a los comandos de control a la vez con diferentes hilos.

Comandos de control

Los comandos de control son los mensajes que se intercambian directamente los pares para controlar todos los aspectos de la videollamada, como señalización de inicio, pausa, final, etc. Los especificados en esta sección son los mínimos a implementar. Cada pareja es libre de diseñar e implementar más, siempre respetando las sintaxis de éstos, para que todos los clientes sean compatibles entre sí (más sobre esto en la sección de protocolos).

Comando CALLING

Objetivo: Señalizar que un nodo quiere establecer una videollamada con otro
Sintaxis: CALLING nick srcUDPport, donde *srcUDPport* es el puerto UDP en el que el llamante desea recibir el video del llamado.
Posibles respuestas/errores:

- CALL_ACCEPTED nick dstUDPport, donde dstUDPport es donde el llamado recibirá el video
- CALL_DENIED nick
- CALL_BUSY, el llamado está en una llamada y no puede recibir la comunicación.

Comando CALL_HOLD
Objetivo: Señalizar que se desea pausar temporalmente una llamada, sin cortarla.
Sintaxis: CALL_HOLD nick
Posibles respuestas/errores: Ninguno

Comando CALL_RESUME
Objetivo: Señalizar que se desea reanudar una llamada anteriormente pausada.
Sintaxis: CALL_RESUME nick
Posibles respuestas/errores: Ninguno

Comando CALL_END
Objetivo: Señalizar que se desea finalizar una llamada.
Sintaxis: CALL_END nick
Posibles respuestas/errores: Ninguno

Gestión del flujo de video

El control de flujo de video es también una parte importante de la práctica, que es donde más libertad se proporcionará. En principio, el objetivo es ser capaz de gestionar situaciones de congestión y diferentes anchos de banda entre los participantes, adaptantado el flujo de video para que la calidad de la videollamada sea la mejor posibles en cada momento. Para ello, se podrá jugar esencialmente con dos parámetros:

- Calidad del video emitido
- Frames por segundo

Ambos parámetros pueden modificarse a través de las funciones adecuadas, que vienen comentadas en el código fuente de partida entregado. Para poder detectar y gestionar una posible congestión, el formato de los paquetes de datos UDP incluirá una cabecera y tendrá el siguiente formato:

- Número de orden#Timestamp#Resolución video#FPS#Datos...

donde:

- *Número de orden* es un identificador del paquete, que la fuente del video incrementa en cada frame, y que servirá al receptor para ordenar, si es necesario, los paquetes desordenados.
- *Timestamp*, en formato UNIX, del momento en el que se envió el paquete desde el origen. Esto permitirá detectar el posible retraso y, si es superior a cierto límite, poner en marcha las medidas anti-congestión.
- *Resolución* del video emitido, en formato simple de cadena. Por ejemplo, "640x480".
- *FPS*: frames per second.
- *Datos*: frame a visualizar en el receptor. Los datos deben ir **comprimidos** para que cada frame pueda caber en un solo datagrama UDP. Para ello usaremos un **códec** muy sencillo en el que cada frame se comprime como una imagen JPEG. Proporcionamos un ejemplo:

```
ret, img = self.cap.read() # lectura de un frame de vídeo

# Compresión JPG al 50% de resolución (se puede variar)
encode_param = [cv2.IMWRITE_JPEG_QUALITY,50]
result,encimg = cv2.imencode('.jpg',img,encode_param)
if result == False: print('Error al codificar imagen')
encimg = encimg.tobytes()

# Los datos "encimg" ya están listos para su envío por la red
#enviar(encimg)

# Descompresión de los datos, una vez recibidos
decimg = cv2.imdecode(np.frombuffer(encimg,np.uint8), 1)

# Conversión de formato para su uso en el GUI
cv2_im = cv2.cvtColor(decimg,cv2.COLOR_BGR2RGB)
img_tk = ImageTk.PhotoImage(Image.fromarray(cv2_im))
... etc
```

Normalmente, este tipo de clientes se codifican utilizando *buffers circulares*, y sería también la solución más adecuada aquí. De esta forma, el receptor del video no comenzará a visualizar el video inmediatamente, sino que lo hará cuando se haya llenado este búfer, y así tenga un margen de seguridad en caso de pequeños cortes. El tamaño del búfer (número de slots) dependerá de cada cliente, pero una regla sencilla para calcularlo puede ser: queremos almacenar 2 segundos de video de margen, a 20 fps = 40 slots.

Protocolos

Para permitir que aquellos estudiantes que así lo deseen puedan diferenciar su práctica añadiéndole mejoras y características extra, el protocolo básico soportará el versionado del mismo.

Así, el protocolo básico descrito hasta el momento tendrá el número de versión V1, que deberá especificarse en el comando REGISTER, y deberá ser soportado por todos los clientes. Si alguna pareja desea añadir nuevas funcionalidades, como un algoritmo mejorado de control de flujo, múltiples flujos de video, cifrado de las comunicaciones, etc., deberá:

1. Escribir una pequeña descripción de la funcionalidad de su protocolo, sintaxis de los mensajes, etc.
2. Contactará y consensuará con su profesor de prácticas la versión definitiva. Una vez hecho esto, se publicará en esta sección para que el resto de estudiantes que así lo quieran puedan también implementarlo. Obviamente, la implementación de estos protocolos extra hará que la puntuación final de la práctica sea superior.

Volver a: Prácticas