Prácticas en Empresa - ARTEC

ANÁLISIS DE LOS FLUJOS DE VÍDEO DE UN SERVIDOR DVR

Servidor de flujos de vídeo DVR

- Conexiones realizadas por los clientes con el servidor de flujos de vídeo DVR.
- Selección de la/s cámara/s que van a transmitir flujos de vídeo al cliente.
- Formatos de vídeo, desde el volcado directo de red al formato *raw* (datos de vídeo H.264).
- Análisis del formato de vídeo propietario, al que llamaré *DHAV*.
- Tratamiento del formato *raw* mediante la herramienta *ffmpeg.*

Conexiones con el DVR

- Se establecen varias conexiones TCP con el servidor, a través de su puerto 37777.
- Estas conexiones son de varios tipos:
  - Primaria (Autenticación y Control de cámaras).
  - Canal sin usar (posiblemente para control PTZ).
  - Canales de vídeo.
- El vídeo se envía en formato *raw* H.264 incrustado en cabeceras de red y propietarias, ambas compuestas de caracteres ASCII.
- Mediante el canal de control se seleccionan las cámaras que enviarán vídeo al cliente.

Conexión primaria

- Sirve para autenticar al usuario y luego para enviar y recibir paquetes de control.
- Se envían primero las credenciales, y al ser validadas se recibe un identificador de sesión.
- Las credenciales se envían como texto plano; tanto usuario como contraseña constan como máximo de 8 caracteres.
- El identificador de sesión se utiliza de nuevo cada vez que se abre una nueva conexión con el servidor DVR.

Conexión primaria - Estructura

- Paquete de conexión (ejemplo):

```
a0 00 00  60 00  00 00  00  uA uB uC uD uE uF uG uH

pA pB pC pD pE pF pG pH  04  01 00  00 00  00 a1 aa
```

  - a0: Indica paquete de autenticación, o de inicio de sesión.
  - uA-uH: Caracteres identificando al usuario.
  - pA-pH: Caracteres de la contraseña.

Conexión primaria - Respuesta del servidor DVR

- Paquete de respuesta (ejemplo):

```
b0  00   00 58 00 00 00 00  00 08 04 08 3d 00 00 00

ID1 ID2 00 00 01 00 00 00  06 00 f9 00  00 04 64 02
```

  - b0: Indica respuesta de autenticación.
  - 58: Es un código de respuesta del servidor.
  - ID1 e ID2: Caracteres con identificador de sesión asociada al cliente de red conectado.

Conexión primaria - Paquetes extra (1)

- Se envían tras recibir ID de sesión. Estáticos.
- Tras enviar cada paquete se recibe un paquete de respuesta por el mismo canal de tamaño 32, 64, 54 y 368. Se envían estos 4 paquetes:

```
a1  00 00 00 00 00 00 00  00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
```

```
a4  00 00 00 00 00 00 00  01 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
```

Conexión primaria - Paquetes extra (2)

```
a4  00 00 00 00 00 00 00  07 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
```

```
a4  00 00 00 00 00 00 00  02 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00   00 00 00 00 00 00 00 00
```

- Después de este proceso es posible enviar y recibir paquetes de control de cámaras por este canal primario.

Conexión secundaria

- Es un canal sin utilizar. Tal vez podría servir para el control de cámaras PTZ, pero no se ha podido comprobar.
- El DVR envía un paquete respuesta y espera.
- Estructura paquete de apertura de canal:

f1 00 00 00 00 00 00 00 ID1 ID2 00 00 02 05 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

  - ID1 e ID2: Caracteres con el identificador de sesión.

Selección de cámaras

- En los clientes analizados que utilizan el SDK, las peticiones para recibir o detener flujos de vídeo siempre tratan de uno en uno.
- No se ha detectado que se abra o cierre la recepción de varios flujos simultáneamente, aunque técnicamente parece posible.
- El cliente indica el mapeado de cámaras solicitadas por el canal primario y recibe aquellos flujos que son indicados en un paquete de apertura por la conexión de vídeo; se asocia un flujo a cada conexión de vídeo abierta.

Conexiones de vídeo (1)

- Se envía un paquete al servidor mediante la conexión primaria, que codifica los canales de vídeo (en nuestro DVR del 1 al 4) que quiere recibir el cliente.
- El cliente abre un nuevo socket para recibir el vídeo de una nueva cámara solicitada.
- Este nuevo socket, al abrirse, envía un paquete al servidor para comenzar a recibir el vídeo (indicando ID de sesión); a partir de entonces el DVR sirve el flujo de manera ininterrumpida.

Conexiones de vídeo (2)

- Cada conexión de vídeo sólo envía el paquete de inicio, y a continuación sólo recibe.
- El resto del tiempo, el hilo de ejecución que controla cada conexión se tiene que encargar de procesar los datos recibidos por red.
- Para cerrar un flujo de vídeo, el método que sigue el SDK proporcionado consiste en enviar un paquete de control por el canal primario y después cerrar directamente la conexión de vídeo asociada.

Paquetes de control de vídeo (1)

- Canal primario (petición cámaras) - 48 bytes:

11	00	00	00	10	00	00	00	CM1	CM2	CM3	CM4
00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00

  - 11...10...: Patrón de control de cámaras.
  - CM1-4: Indica los flujos que el cliente desea recibir; 01 → activado; 00 → desactivado.
- Si el DVR tiene más cámaras, deberían estar mapeadas posteriormente a las indicadas.

Paquetes de control de vídeo (2)

- Apertura por la conexión de vídeo:

```
f1  00 00 00 00 00 00 00  ID1 ID2 00 00 01 CM 00 00

00 00 00 00 00 00 00 00  00   00  00 00 00 00 00 00
```

  - f1: Código enviado para recibir vídeo de cam. CM.
  - ID1 e ID2: Identificador de sesión.
  - CM: Indica la cámara, del 1 al 4.
- Después de este paquete, la conexión de vídeo recibirá continuamente cabeceras de red y bloques propietarios *DHAV*, con el vídeo *raw* incrustado en éstos.

Cierre de conexiones - SDK

- Para cerrar todas las conexiones de vídeo, el SDK envía un paquete de control para cada conexión abierta, indicando sucesivamente el cierre de uno de los flujos abiertos.
- La conexión secundaria sólo se cierra antes de terminar la sesión.
- Paquete para cerrar la sesión (canal primario):

```
0a  00 00 00 00 00 00 00  ID1 ID2 00 00 00 00 00 00

00 00 00 00 00 00 00 00   00  00  00 00 00 00 00 00
```

  - 0a: Fin de sesión. ID1 e ID2: Identificador de sesión.

Formato de los datos de vídeo

- El formato captado por las cámaras parece ser encapsulado en *raw* H.264.
- El flujo *raw* de las cámaras se particiona, y cada bloque se inserta entre cabeceras y colas propietarias en formato *DHAV.*
- Antes de cada bloque *DHAV* se inserta una cabecera de red que incorpora datos para identificar al siguiente bloque *DHAV*.
- Los bloques *DHAV* no son siempre del mismo tamaño.

Cabeceras de red

- Antes de cada bloque *DHAV* se inserta una cabecera de red de 32 bytes, que incorpora unos delimitadores que utilizarán la cabecera y la cola *DHAV* que le seguirán.
- Formato:

```
bc  00 00 00 DL1 DL2 00 00  CM 00 00 00 00 00 00 00

00 00 00 00   00   00   00 00   00 00 00 00 00 00 00 00
```

  - bc: Indica cabecera de red.
  - DL1 y DL2: Delimitadores usados por el bloque *DHAV*.
  - CM: Cámara a la que pertenece el flujo.

Formato propietario *DHAV*

- Este es el formato que utiliza el DVR para almacenar grabaciones y reproducirlas con SDK. Consiste en bloques con cabeceras de 32 bytes (o caracteres) y colas de 8 bytes.
- Los datos *raw* se encuentran entre la cabecera y la cola.
- El DVR manda bloques *DHAV* a clientes de red, añadiendo cabeceras de red antes de cada bloque *DHAV*.
- Además, envía intercalados bloques *DHAV* con datos *raw* y bloques *DHAV* vacíos (todo ceros).

Cabeceras propietarias *DHAV* (1)

- El DVR guarda grabaciones en este formato, en archivos con extensión *'.dav'*.
- Formato de la cabecera *DHAV* (32 bytes):

D H A V TY 00 00 00 EE EE CC 00 DL1 DL2 00 00

16 bytes con función sin identificar, parecen random

  - D, H, A, V: Enteros correspondientes a los caracteres ASCII de 'D', 'H', 'A' y 'V'.
  - TY: Tipo de bloque; f0 → bloque vacío, otro (fd o fc) → bloque con datos *raw*.
  - DL1 y DL2: Delimitadores para localizar la cola.
  - EE y CC: Contadores de cabeceras *DHAV*.

Cabeceras propietarias *DHAV* (2)

- Los bloques *DHAV* no tienen un tamaño fijo, normalmente son grandes pero a veces son muy pequeños. Los bloques vacíos tienen todo ceros entre cabecera y cola, y pueden ser muy grandes también.
- Para encontrar un final al bloque *DHAV* se busca la cola *DHAV* (8 bytes)que tiene este formato: `d h a v DL1 DL2 00 00`
  - d, h, a, v: Enteros correspondientes a los caracteres ASCII de 'd', 'h', 'a' y 'v'.
  - DL1 y DL2: Delimitadores pertenecientes a cabecera y cola *DHAV*.

Formato *raw* H.264

- Una vez extraidas todas las cabeceras, de red y *DHAV* con y sin datos, los datos resultantes son vídeo en formato *raw* H.264.
- No he encontrado ningún reproductor de vídeo capaz de reconocer este formato *raw*, ni el *vlc*.
- Sin embargo, hay aplicaciones capaces de reconocer *raw* y traducirlo a otros formatos más faciles de reproducir.
- Para ello, una versátil herramienta disponible es *ffmpeg*.

Herramienta *ffmpeg*

- *ffmpeg* dispone de multitud de opciones de conversión, soportando la lectura y escritura de una amplia gama de formatos.
- Los formatos soportados por plataformas Android son *'.mp4'* y *'.3gp'*, ambos también soportados por *ffmpeg*.
- Una vez volcado el vídeo *raw* de una cámara en un archivo (camX.raw), el comando utilizado para convertir a uno u otro formato es:
  - ffmpeg -i camX.raw -f mp4 camX.mp4
  - ffmpeg -i camX.raw -f 3gp camX.3gp

Estructura de la solución implementada

- Hilo principal para gestionar peticiones del usuario.
- Un hilo de ejecución para cada conexión abierta con el servidor.
- Procesamiento en cada hilo de vídeo de los datos que llegan desde la red.
- Volcado en fichero del formato deseado por el usuario (*raw* o *DHAV*) según la elección al inicio del hilo principal, realizado durante el procesamiento de los hilos de vídeo.

Secuencia de la solución

- Login por defecto en el DVR de pruebas.
- Selección del formato de vídeo a volcar.
- Apertura de canales de vídeo.
- Cierre de canales y/o programa.
- Conversión a otros formatos mediante *ffmpeg*.
- Visionado de los vídeos en diferido mediante algún reproductor (*vlc* por ejemplo).
- Nota: Si abrimos varias veces una misma cámara, los nuevos vídeos sobreescribirán los antiguos.

Mejoras para aplicación de vídeo en directo

- Añadir campos para selección de host y puerto.
- Enlazar el vídeo *raw* extraído con una conversión dinámica para volcar en buffer.
- Implementar GUI con reproductor de vídeo para visionar la conversión desde buffer.
- Defecto: Para ver un vídeo perfecto es necesario contabilizar las diferentes cabeceras/colas y conocer el final de cada bloque *DHAV*, para poder reubicar los nuevos paquetes al principio del buffer de lectura de red, ya que los bloques no son de tamaño fijo.

Conclusiones

- El formato propietario podría estar pensado para ofrecer funcionalidades al servicio del usuario, pero viendo los bloques vacíos parece un enmascaramiento del formato *raw* para mantener una dependencia del SW propietario.
- La mayor dificultad en conseguir una aplicación de vídeo en directo radica en el proceso de conversión *raw* a otro formato, pues sería necesario el código para su implementación, el uso de una librería que no ha sido encontrada, o el entubamiento de E/S de procesos (hilos de vídeo → *raw* → *ffmpeg* → GUI).