Universidad Politécnica de Cartagena

Escuela Técnica Superior de Ingeniería de Telecomunicación

LABORATORIO DE CONTENIDOS DIGITALES

Práctica 5: Video Conferencia usando comunicaciones Multicast UDP.

Profesor:

Antonio Javier García Sánchez.

En esta práctica se va a implementar una **Video Conferencia usando comunicaciones Multicast UDP**. La funcionalidad de nuestra aplicación se resume en los siguientes puntos:

- Un Servidor de Video captura imágenes y las difunde a través de un grupo Multicast.
- Los mensajes que forman las imágenes se envían vía UDP a cada uno de los clientes.
- Cada cliente del del grupo Multicast recibe los mensajes, forma la imagen y la presenta al usuario.

Esta aplicación necesita utilizar la librería *DirectShow* para el tratamiento y envío de la imagen en tiempo real. En la versión de Visual Studio 2013, esta librería se encuentra incluida.

La figura 1 representa el esquema a implementar por el alumno :

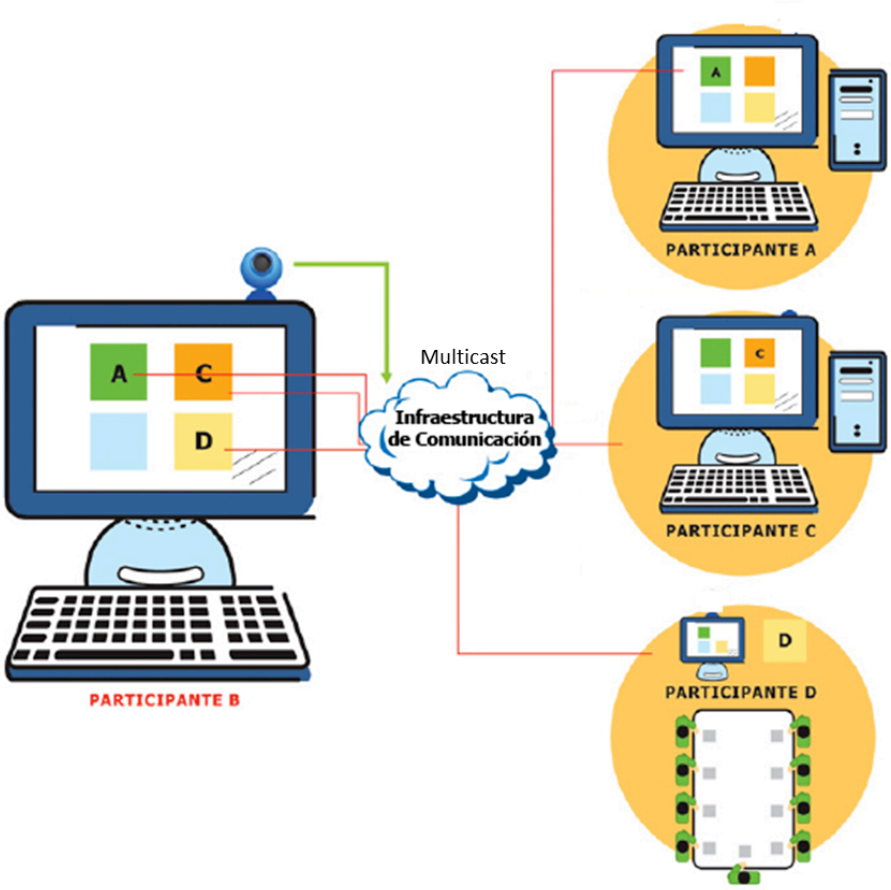


Figura 1. Video Conferencia Multicast UDP

1.- Video Streaming Server

Para la realización de la estación servidora necesitamos crear una aplicación que obtenga la imagen en tiempo real de la cámara web, se conecte al grupo multicast y envíe mediante protocolo UDP las imágenes que se vayan obteniendo de la cámara.

A) Captura de Video a través de una WebCam

La *using a* añadir es la siguiente: using System.Drawing.Imaging.

En primer lugar, el alumno deberá detectar el número y características de las cámaras web que el PC posee:

```
Camera cam in CameraService.AvailableCameras
```

Una vez seleccionada la cámara (por ejemplo en un *comboBox*), se debe iniciar la captura de imágenes por la cámara web. Para ello se establece diversos parámetros como el tamaño de la imagen capturada o el número de imágenes por segundo.

```
private CameraFrameSource _frameSource;
Camera c = (Camera)comboBoxCameras.SelectedItem;
setFrameSource(new CameraFrameSource(c));
_frameSource.Camera.CaptureWidth = 320;
_frameSource.Camera.CaptureHeight = 240;
_frameSource.Camera.Fps = 20;
_frameSource.NewFrame += OnImageCaptured;
```

Con esta última línea nos garantizamos que se capture una imagen de la cámara web. El código del método es el siguiente:

```
public void OnImageCaptured(Touchless.Vision.Contracts.IFrameSource
frameSource, Touchless.Vision.Contracts.Frame frame, double fps)
        {
            _latestFrame = frame.Image;
            pictureBoxDisplay.Invalidate();
        }
```

Con el manejador de eventos *PaintEventHandler* creamos un método *drawLatestImage,* el cual es el encargado de generar una imagen *Bitmap*. Esto es esencial para poder: (i) visualizar la imagen y (ii) poder enviarla en modo Multicast.

```
private static Bitmap _latestFrame;
private void drawLatestImage(object sender, PaintEventArgs e)
       {
           if (_latestFrame != null)
           {
e.Graphics.DrawImage(_latestFrame, 0, 0, _latestFrame.Width,
_latestFrame.Height);
....
       }
```

La imagen es visualizada a través del elemento *pictureBoxDisplay*

```
pictureBoxDisplay.Paint += new PaintEventHandler(drawLatestImage);
_frameSource.StartFrameCapture();
```

B) Comunicaciones Multicast UDP

Los requisititos que debe implementar son:

- Añadirse al grupo multicast.
- Enviar imágenes en formato JPEG al cliente.

B.1 Crear un grupo multicast.

Los *using* que habrá que añadir son:

```
using System.Net;
using System.Net.Sockets;
using System.IO;
```

Para crear un grupo multicast se va a utilizar el objeto *UdpClient*. *UdpClient* proporciona servicios de red mediante el protocolo de datagramas de usuarios (UDP).

Dentro de sus métodos se selecciona *JoinMulticastGroup*, que internamente proporciona la creación y adhesión al grupo multicast:

```
UdpClient udpserver = new UdpClient();
IPAddress multicastaddress=IPAddress.Parse("224.0.0.1");
udpserver.JoinMulticastGroup(multicastaddress);
```

IPEndPoint representa un punto final de red como una dirección IP y un número de puerto. En el caso servidor se inicia dicho punto final con la dirección Multicast y puerto del cliente de video.

```
IPEndPoint remote=new IPEndPoint(multicastaddress, UDP_PORT);
```

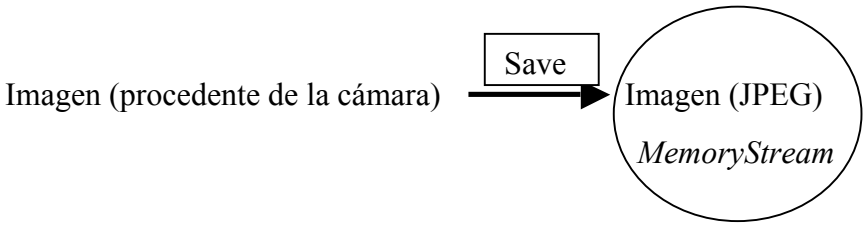
B.2 Envío de imágenes.

La aplicación Video Server envía los mensajes al Cliente usando el método *Send* de *UdpClient.* Se envía un datagrama al extremo remoto con la siguiente sintaxis:

```
public int Send(byte[], int, IPEndPoint);
udpServer.Send(buffer, buffer.Length, remote);
```

En el argumento *buffer* contiene la imagen en formato JPEG. Este argumento solo maneja **array de bytes** por tanto habrá que convertir la imagen en formato JPEG en este tipo. Para ello se puede usar el método *Toarray()*.

Otro aspecto a tener en cuenta es como manejar cada imagen de video que captura la cámara y convertirlo en fotogramas JPEG. En primer lugar el alumno se familiarizará con la clase *MemoryStream*, la cual hace de manejador de datos streaming creando un objeto. Este objeto es el que utiliza para convertir el conjunto de datos que forman la imagen en el formato JPEG. Para ello el alumno utilizará el método *Save* ya visto en la práctica del "Convertidor de Imágenes" y lo ejecutará sobre la imagen capturada de la cámara. El siguiente esquema resume el proceso indicado:



2.- Cliente de vídeo.

La aplicación cliente al iniciarse se debe conectar al grupo multicast, obtener la imagen enviada por el servidor y mostrarla por pantalla. Todo ello de forma recurrente hasta que el servidor deje de enviar imágenes.

A) Comunicaciones Multicast UDP

Se utiliza como en el caso servidor la clase *UdpClient*, teniendo ahora en cuenta que hay que añadirse y enviar los mensajes a dicho grupo. Para ello además de los pasos realizados en el servidor se utiliza la clase *IPEndPoint* de la siguiente manera:

```
IPEndPoint remoteep = new IPEndPoint(IPAddress.Any, UDP_PORT );
```

Es decir, el cliente toma su IP y recibirá los datos en el puerto designado.

Además para que el cliente pueda aceptar varias conexiones, el alumno deberá implementar los métodos *Client.SetSocketOption* y *Client.Bind* de la clase *UdpClient*.

El cliente se queda bloqueado hasta recibir datos de imágenes de la aplicación Video Server. Se usa para ello el método *Receive* dentro de la clase *UdpClient* que recibe un array de bytes.

```
Byte[] buffer = UdpCliente.Receive(ref localEp);
```

Estos datos son manejados por la clase *MemoryStream* con el objeto de poder convertirlos en una imagen. Para ello, una vez creado el objeto *MemoryStream* el cual debe contener los datos de la imagen, se usa el método *FromStream* de la clase *Image.* Automáticamente .NET detecta que la imagen está en formato es JPEG.

B) Visualización de imágenes

Simplemente con el método *Image* de *pictureBoxDisplay,* se podrá visualizar las imágenes en formato JPEG.

NOTA: Como se ha comentado la aplicación servidora queda bloqueada recibiendo los datos de los clientes, provocando que el entorno gráfico no se ejecute. Para solucionar este problema se va implementar tasks gestionado por el CLR, el cual decide si la aplicación debe utilizar hilos de ejecución, cuantos y cuándo serán lanzados. Un ejemplo lo tenemos en el siguiente bloque de código:

```
Task t1 = new Task(visualizar_imagen);
```

```
t1.Start();
private void visualizar_imagen()
        {
            while (true)
            {
                try
                {
                …….
               }
```