在本练习中，我们将使用 UDP 组播通信实现视频会议。应用程序的功能概述如下：
- 视频服务器捕获图像并通过组播组进行广播。
- 组成图像的信息通过 UDP 发送给每个客户端。
- 组播组中的每个客户端接收信息，形成图像并呈现给用户。

该应用程序需要使用 DirectShow 库来实时处理和发送图像。Visual Studio 2017 版本中包含了该库。
下图是学生要实现的模式：

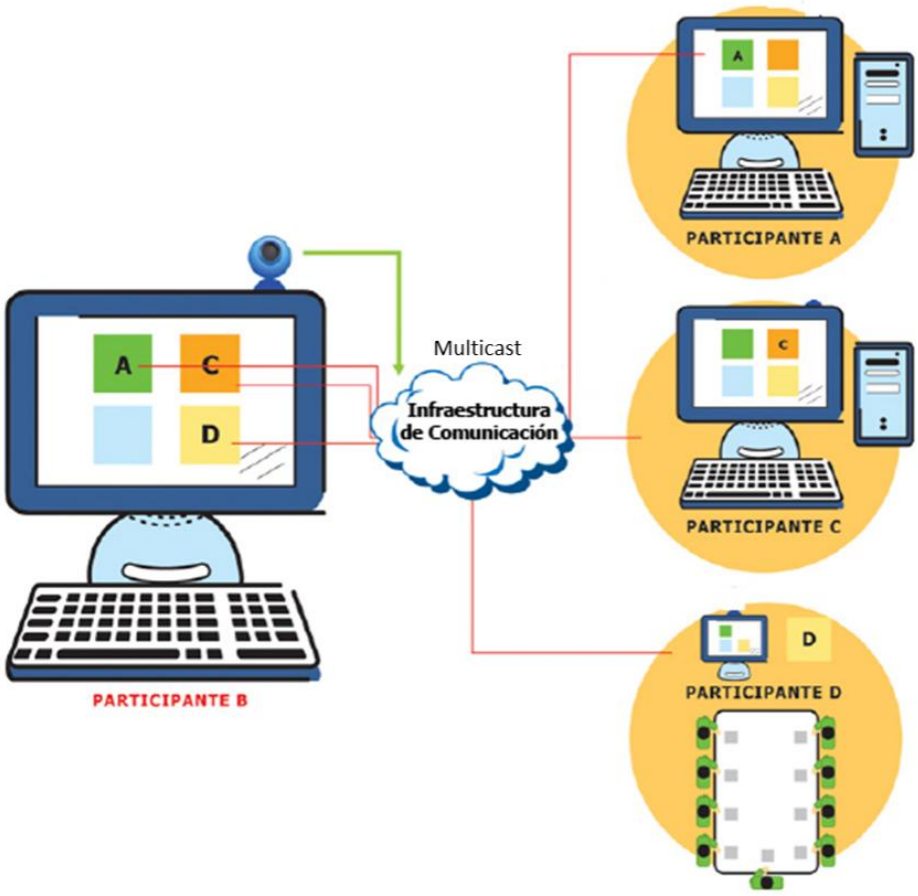


图 UDP 组播视频会议

1.- 视频流服务器

为了实现服务器站，我们需要创建一个应用程序，从网络摄像头实时获取图像，连接到多播组，并通过 UDP 协议发送从摄像头获取的图像。

A) 通过网络摄像头采集视频

要添加的 using 如下：using System.Drawing.Imaging.XML。

首先，学生必须检测 PC 上网络摄像头的数量和特性：

```
foreach(Camera cam in CameraService.AvailableCameras)
```

应添加以下全局变量

```
private CameraFrameSource _frameSource;
private static Bitmap _latestFrame;
```

一旦选择了摄像头（例如在 comboBox 中并使用上述 foreach 指令），就必须启动网络摄像头捕捉图像。为此，需要设置各种参数，如捕获图像的大小或每秒图像的数量。

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

图片通过 pictureBoxDisplay 元素显示

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

通过最后一行，我们可以确保从网络摄像头捕捉到图像。该方法的代码如下

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

通过 PaintEventHandler 事件处理程序，我们创建了一个 drawLatestImage 方法，该方法负责生成一个位图图像。这对于 (i) 显示图像和 (ii) 以组播模式发送图像至关重要

```
private static Bitmap _latestFrame;
private void drawLatestImage(object sender, PaintEventArgs e)
{
if (_latestFrame != null)
{ // 此处 _latestFrame 图像的大小应调整为 320x240
e.Graphics.DrawImage(_latestFrame, 0, 0, _latestFrame.Width,
_latestFrame.Height);
//发送图像的代码将插入此处
}
```

要调整图像大小，可以生成另一个新的、未指定大小的位图：

```
Bitmap(bitmap_old, new Size(new_width, new_height));
```

B) UDP 多播通信

您必须满足的要求有

- 添加到多播组。
- 向客户端发送 JPEG 格式的图像。

B.1 创建一个组播组。

要添加的使用是

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

要创建组播组，将使用 UdpClient 对象。UdpClient 使用用户数据报协议（UDP）提供网络服务。

在其方法中，选择了 JoinMulticastGroup，它在内部提供了创建和加入组播组的功能：

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

IPEndPoint 用 IP 地址和端口号表示网络端点。在服务器情况下，该端点用视频客户端的组播地址和端口初始化

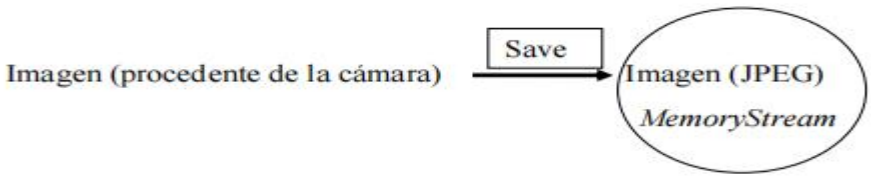
```
IPEndPoint remote=new IPEndPoint(multicastaddress, UDP_PORT);
```

B.2 发送图像

视频服务器应用程序使用 UdpClient 的发送方法向客户端发送信息。向远端发送的数据报语法如下：

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

另一个需要考虑的方面是如何处理摄像机捕获的每幅视频图像并将其转换为 JPEG 帧。首先，学生将熟悉 MemoryStream 类，该类通过创建对象充当流数据处理器。该对象用于将形成图像的数据集转换为 JPEG 格式。为此，学生将使用在 "图像转换器 "练习中已见过的保存方法，并在从摄像机捕获的图像上执行该方法，将 MemoryStream 类型的对象传给它，并指示转换为 JPEG 格式。下图概括了所述过程：



参数缓冲区包含 JPEG 格式的图像。该参数只能处理字节数组，因此必须将 JPEG 格式的图像转换为这种类型。为此，可以使用 MemoryStream 类型对象的 Toarray() 方法。

2.- 视频客户端。

客户端程序启动后，必须连接到多播组，获取服务器发送的图像并将其显示在屏幕上。所有这些都要反复进行，直到服务器停止发送图像。

A) UDP 多播通信

与服务器的情况一样，使用 UdpClient 类，但现在要考虑到必须将信息添加并发送到该组。为此，除了在服务器中执行的步骤外，还按以下方式使用 IPEndPoint 类：

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

也就是说，客户端使用其 IP，并在指定端口接收数据。

此外，为了使客户端能够接受多个连接，学生必须实现 UdpClient 类的 Client.SetSocketOption 和 Client.Bind 方法。

在从视频服务器应用程序接收图像数据之前，客户端将被阻塞。UdpClient 类中的 Receive 方法用于接收字节数组。

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

这些数据由 MemoryStream 类处理，以便将其转换为图像。为此，一旦创建了 MemoryStream 对象（其中必须包含 Save MemoryStream 图像数据），就会使用 Image 类的 FromStream 方法。.NET 会自动检测到图像是 JPEG 格式的。

B) 图像显示

只需使用 pictureBoxDisplay 的 Image 方法，就能以 JPEG 格式显示图像。

注意：如前所述，服务器应用程序在接收来自客户端的数据时会受阻，导致图形环境无法运行。为解决这一问题，将执行由 CLR 管理的任务，由 CLR 决定应用程序是否应使用执行线程、线程的数量和启动时间。下面的代码块给出了一个例子：

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

3.- 评估视频服务性能。

在项目的第二部分，学生需要发送嵌入通信协议的视频数据包。为此，他们将设计一个标头，至少包含图像编号、在该图像中接收到的数据包的序列号和时间戳等字段。每个数据包应由这些字段和从 MemoryStream 类型对象的 ImageToByteArray(_latestFrame) 方法（尤其是 Byte[] 缓冲区）中获得的视频信息有效载荷组成。学生应

1）在发送方，获取最佳有效载荷大小，以便正确显示图像序列。
2）在接收方，一方面要读取报头，以计算延迟和抖动指标以及丢失数据包的数量。在这种情况下，图形演示将是一个很好的工具。另一方面，学生必须将 JPEG 图像与数据包有效载荷组合在一起。
3）将对学生将 RTP 协议作为通信协议实施的情况进行评估。

为观察成绩的变化，将对附件 1 和附件 2 的包含情况进行评估。

附件 1. UDP 多播聊天
1.- 服务器应用

要求如下：
- 创建多播组。
- 从添加到组播组的所有客户端接收数据。
- 在应用程序中显示客户端发送的信息。

1.1 创建组播组

要添加的使用是

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

要创建组播组，将使用 UdpClient 对象。UdpClient 使用用户数据报协议（UDP）提供网络服务。

在其方法中，选择了 JoinMulticastGroup，它在内部提供了创建和加入组播组的功能：

```
UdpClient udpclient = new UdpClient(8080);
IPAddress multicastaddress=IPAddress.Parse("224.0.0.1");
udpclient.JoinMulticastGroup(multicastaddress);
```

IPEndPoint 用 IP 地址和端口号表示网络端点。在服务器情况下，它不必与任何其他服务器绑定，引用为空

```
IPEndPoint remote=null;
```

1.2 接收数据

服务器在接收到来自客户端的数据之前是阻塞的。UdpClient 类中的 Receive 方法用于接收字节数组。

1.3 数据表示

来自不同客户端的文本必须打印在屏幕上。使用 ListBox 和 Textbox 来表示：

- ListBox 包含所有已到达信息的历史记录。
- 文本框，包含最后收到的信息。

服务器通过 Receive 方法接收字节数组。ListBox 和 Textbox 组件支持字符串。在.Net 中，通过使用编码类，可以在不使用指针或字符数组的情况下传递字符串。Encoding 表示字符编码。该类使用 Unicode 类型，它将每个字符编码为两个连续字节。

注：如上所述，服务器应用程序在接收来自客户端的数据时会被阻塞，导致图形环境无法执行。为解决这一问题，将使用接收客户端数据的代码来执行线程。

线程的实现是通过线程类来完成的，添加以下使用

```
using System.Threading;
```

Un ejemplo sería el siguiente:

```
Thread t = new Thread(new ThreadStart(MyThreadMethod));
t.Start();
private void MyThreadMethod()
{
while(true)
{
.
//Código de recepción de datos
.
}
```

2.- 客户应用程序

需要实现的要求如下

- 添加到多播组。
- 向客户端发送信息。

2.1 加入组播组

与服务器的情况一样，使用 UdpClient 类，现在考虑到有必要添加和发送信息到该组。为此，除了在服务器中执行的步骤外，还要按以下方式使用 IPEndPoint 类：

IPEndPoint `remoteep = new IPEndPoint( multicastaddress, 8080 );`

2.2 图形界面

我们将使用 RichTexBox 组件，它允许用户在屏幕上书写文本。这些文本是要发送的字符串。我们将创建一个按钮，将信息发送到服务器。

2.3. 发送信息

客户端使用 UdpClient 的发送方法向服务器发送信息。向远端发送的数据报语法如下：

```
public int Send(byte[], int, IPEndPoint);
```

附件 2. 使用 A-law 编码的音频传输

本次实践的目的是使用 A-law 编码实现音频传输。我们的应用功能如下：

- 使用 UDP 作为通信协议和 A-law 作为音频编码器，通过麦克风向远程主机传输音频。
- 通过扬声器接收和播放音频。

图 2 表示要创建的应用程序：

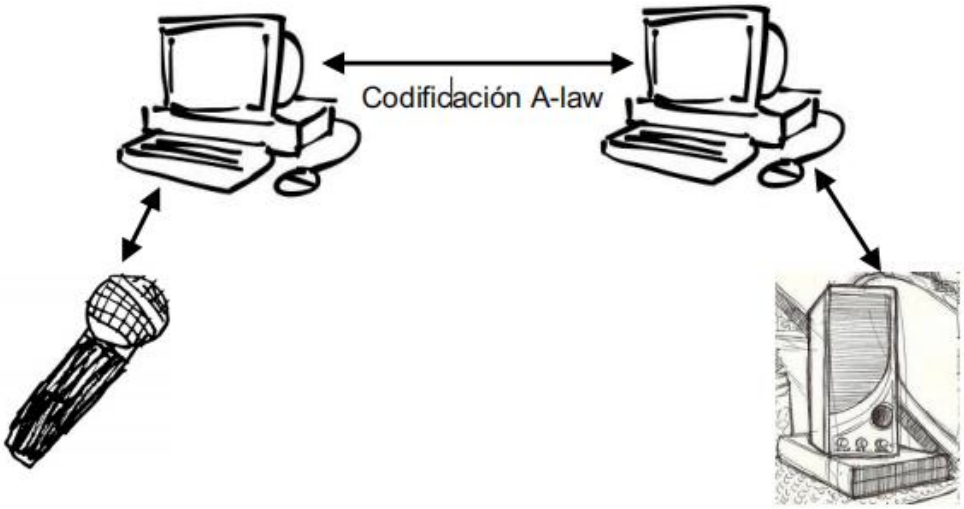


Figura 2. Transmisión Audio

1.音频传输/接收

作为本节的一般性说明，将加载以下使用说明：

```
using ALaw;
using NAudio.Wave;
using NAudio.CoreAudioApi;
```

注意：必须安装 Naudio 软件包

1.1 设置数据

A-law 编码

首先，将使用 waveformat 类设置音频波的格式和属性。学生必须了解该类的以下成员：Channels、FormatTag、SamplesPerSecond、BitsPerSample、BlockAlign、AverageBytesPerSecond。

在设计上述成员时，建议使用以下参数：

- short channels = 1; //Stereo.
- short bitsPerSample = 16; //16Bits
- int samplesPerSecond = 22050; //22KHz。
- PCM 编码

为此可执行以下函数：

```
private void listBox1_SelectedIndexChanged(object sender,
EventArgs e)
{
WaveIn waveIn = new WaveIn();
int inputDeviceNumber = listBox1.SelectedIndex;
waveIn.BufferMilliseconds = 50;
waveIn.DeviceNumber = inputDeviceNumber;
waveIn.WaveFormat = new WaveFormat(8000, 16, 1);
waveIn.DataAvailable += OnAudioCaptured;
waveIn.StartRecording();
}
```

1.2 从麦克风采集音频

```
void OnAudioCaptured(object sender, WaveInEventArgs e)
{
byte[] encoded = ALawEncoder.ALawEncode(e.Buffer);
udpSender.Send(msg, msg.Length, endPoint);
}
```

这将与实现 A-law 编码和远程发送一样简单。

注 1：A-law 编码器代码将由教师提供

注 2：A-law 编码器可将缓冲区大小减半

1.3 扬声器音频播放

在音频播放器中，必须使用音频捕获端指示的值创建 WaveOut 和 Waveformat 对象

```
waveOut = new WaveOut();
waveProvider = new BufferedWaveProvider(new
WaveFormat(8000, 16, 1));
waveProvider.DiscardOnBufferOverflow = true;Y para
reproducer en el altavoz se puede implementar una
función tal que así:
private void ListenerThread(object state)
{
var endPoint = (IPEndPoint)state;
try
{
waveOut.Init(waveProvider);
waveOut.Play();
while (listening)
{
byte[]
b = clientAudio.Receive(ref
endPoint);
byte[] payload = deserializeHeader(b,
false); // 由学生实施
short[]
decoded =
ALawDecoder.ALawDecode(payload);
byte[] result = new byte[decoded.Length
* 2];
Buffer.BlockCopy(decoded, 0, result, 0,
result.Length);
waveProvider.AddSamples(result, 0,
result.Length);
}
}
catch (SocketException)
{
}
}
```

注：A-law 音频解码代码将由教师提供。