Pregunta 2

Indique hasta donde ha implementado:

1) Protocolo (Indicar si es RTP)

2) Chat Multicast

3) Audio

4) Evaluación de Prestaciones

He implementado las cuatro funciones
(1) Implementación del protocolo RTP para encapsular y enviar paquetes de audio y vídeo

Pregunta 3

Indique hasta donde ejecuta correctamente su programa:

1) Protocolo

2) Chat Multicast

3) Audio

4) Evaluación de Prestaciones

1. Protocolo (protocolo) Protocolo RTP:

El proyecto implementa el protocolo RTP para encapsular y transmitir datos de vídeo y audio. Tanto el servidor como el cliente utilizan el protocolo RTP para transmitir y recibir vídeo y audio.

2. Multidifusión de chat

Chat Multicast: El proyecto implementa la capacidad de enviar y recibir mensajes de chat a través de UDP multicast. Los usuarios pueden enviar mensajes de chat, que serán recibidos y visualizados por todos los clientes que se unan al grupo multicast.

3. Audio (Audio) Captura y Transferencia de Audio:

El proyecto implementa la funcionalidad de capturar datos de audio, codificarlos (utilizando codificación A-law) y enviar datos de audio a través de UDP multicast. El servidor es capaz de capturar audio y enviarlo a través del protocolo RTP, y el cliente es capaz de recibir datos de audio y decodificarlos y reproducirlos.

4. Evaluación de Prestaciones

Evaluación de Prestaciones: El proyecto implementa una evaluación de prestaciones de la transmisión de vídeo y audio. Se calculan métricas de rendimiento como el número de paquetes enviados y recibidos (paq/s), el retardo (delay) y el jitter (jitter) y

se muestran periódicamente a través de temporizadores. Esta información se muestra al usuario en la interfaz.

Pregunta 4

Según el código que ha desarrollado, comente los detalles de implementación de:

1) En el servidor de vídeo: la fragmentación de la imagen en paquetes, los campos de la cabecera implicados, y la formación del mensaje enviado al receptor

corte de imágenes

```
1 个引用
private void drawLatestImage(object sender, PaintEventArgs e)
{
    if (_latestFrame != null)
    {
        _latestFrame = new Bitmap(_latestFrame, new Size(320, 240));
        e.Graphics.DrawImage(_latestFrame, 0, 0, _latestFrame.Width, _latestFrame.Height);

        UdpClient udpServer = new UdpClient();
        IPAddress multicastaddress = IPAddress.Parse("224.0.0.4");
        udpServer.JoinMulticastGroup(multicastaddress);

        IPEndPoint remote = new IPEndPoint(multicastaddress, 8080);

        Byte[] buffer = ImageToByteArray(_latestFrame);
        udpServer.Send(buffer, buffer.Length, remote);

        if (button4.Enabled == false && button2.Enabled == false)
        {
            // transmit the image
            // compress the frame
            Bitmap cFrame = new Bitmap(_latestFrame, new Size(320, 240));

            // Conversion to JPEG
            jpegFrame = new MemoryStream();
            cFrame.Save(jpegFrame, ImageFormat.Jpeg);

            // send through the channel
            video.sendJPEG(jpegFrame);
            num_video++;
        }

    }
}
```

La imagen se captura y se comprime en formato JPEG y, a continuación, se llama al método sendJPEG para enviarla a través del protocolo RTP

El método sendJPEG en la clase RTP

Se encarga de trocear la imagen en paquetes y enviarlos a la aplicación

```
public String sendJPEG(MemoryStream frame)
{
    // Enviamos imagen por el canal (JPEG)
    // Payload type == 26 -> JPEG
    byte[] toSend = newPacket(frame.ToArray(), sequence, 26);

    try
    {
        client.Send(toSend, toSend.Length, remote);
        sequence++;
    }
    catch (Exception)
    {
        return "KO";
    }
    return "OK";
}
```

El método principal para construir paquetes RTP es newPacket:

```
private byte[] newPacket(byte[] data, int nSeq, int mPayloadType)
{
    switch (mPayloadType)
    {
        // Audio 8kHz
        case 0:
            timestamp_interval = 160;
            break;
        // Audio 8kHz
        case 8:
            timestamp_interval = 160;
            break;
        // Video 90kHz
        case 26:
            timestamp_interval = 1800;
            break;
    }

    uint timestamp = (uint)(nSeq * timestamp_interval);

    header = createHeader(nSeq, timestamp, mPayloadType);
    payload = new byte[data.Length];
    payload = data;

    packet = new byte[data.Length + header.Length];

    for (int i = 0; i < header.Length; i++)
    {
        packet[i] = header[i];
    }

    for (int j = 12; j < packet.Length; j++)
    {
        packet[j] = payload[j - 12];
    }

    return packet;
}
```

El método createHeader para crear una cabecera RTP:

```
1 个引用
private byte[] createHeader(int nSeq, uint mTimestamp, int mPayloadType)
{
    if(this.sequence >= 65535)
    {
        nSeq = 0;
        this.sequence = 0;
    }

    int version = 2;
    int padding = 0;
    int extension = 0;
    int csrcCount = 0;
    int marker = 0;
    int payloadType = mPayloadType;
    int sequence = nSeq;
    uint timestamp = mTimestamp;
    long SSRC = 0;

    byte[] buf = new byte[12];

    // Assembling according the spec
    // Byte 1.
    buf[0] = (byte)((version & 0x3) << 6 | (padding & 0x1) << 5 | (extension & 0x0) << 4 | (csrcCount & 0x0));

    // Byte 2.
    buf[1] = (byte)((marker & 0x1) << 7 | payloadType & 0x7f);

    // Byte 3 y 4. Numero de secuencia. MSB + LSB. Big endian
    buf[2] = (byte)((sequence & 0xff00) >> 8);
    buf[3] = (byte)(sequence & 0x00ff);

    // Timestamp on 4 bytes. Big endian
    buf[4] = (byte)((timestamp & 0xff000000) >> 24);
    buf[5] = (byte)((timestamp & 0x00ff0000) >> 16);
    buf[6] = (byte)((timestamp & 0x0000ff00) >> 8);
    buf[7] = (byte)(timestamp & 0x000000ff);

    // SSRC
    buf[8] = (byte)((SSRC & 0xff000000) >> 24);
    buf[9] = (byte)((SSRC & 0x00ff0000) >> 16);
    buf[10] = (byte)((SSRC & 0x0000ff00) >> 8);
    buf[11] = (byte)(SSRC & 0x000000ff);

    // Devolvemos todo el header
    return buf;
}
```

2) En el receptor de vídeo: Descomposición del mensaje en los campos de cabecera y payload. Composición de los paquetes para generar la imagen.

En el código cliente, la parte de recepción y ensamblaje del paquete de vídeo se encuentra principalmente en el método VideoThread:

Descomponer el mensaje

```
private void VideoThread()
{
    while (true)
    {
        try
        {
            byte[] receivedData = videoclient.Receive(ref videoremote);

            // Convert to image
            byte[] imageBytes = GetRtpPayload(receivedData);
            num_video++;

            // Save Serial Number
            int sequenceNumber = GetRtpSequenceNumber(receivedData);
            sequence_video.Add(sequenceNumber);

            // Get timestamp
            long currentTime = DateTimeOffset.Now.ToUnixTimeSeconds();
            int timestamp = GetRtpTimestamp(receivedData);

            // Computational latency and jitter
            if (video_prev_timestamp == 0)
            {
                video_jitter = 0;
            }
            else
            {
                video_delay = (currentTime - video_prev_time) - (timestamp * 0.00001 - video_prev_timestamp * 0.00001);
                video_jitter = video_jitter + (Math.Abs(video_delay) - video_jitter) / 16;
            }

            // Storing information about timestamps and times
            video_prev_timestamp = timestamp;
            video_prev_time = currentTime;

            // Update Tags
            UpdateLabel(label7, $"Delay: {video_delay:0.00} ms");
            UpdateLabel(label8, $"Jitter: {video_jitter:0.00} ms");

            Image frameImage = Image.FromStream(new MemoryStream(imageBytes));
            UpdatePictureBox(pictureBox1, frameImage);

            // Marked as receiving RTP video
            UpdateCheckBox(checkBox2, true);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error receiving video: {ex.Message}");
        }
    }
}
```

Aquí, el paquete recibido se descompone en cabecera RTP y carga. La carga se extrae y se convierte en una imagen. La información de la cabecera se utiliza para calcular el retardo y la fluctuación de fase y actualizar las etiquetas UI correspondientes.

Obtención de cargas RTP, números de secuencia y marcas de tiempo

```
2 个引用
private byte[] GetRtpPayload(byte[] rtpPacket)
{
    // Assuming an RTP header length of 12 bytes
    const int headerLength = 12;
    byte[] payload = new byte[rtpPacket.Length - headerLength];
    Array.Copy(rtpPacket, headerLength, payload, 0, payload.Length);
    return payload;
}

2 个引用
private int GetRtpSequenceNumber(byte[] rtpPacket)
{
    // Sequence number in RTP header at 2-3 bytes
    return (rtpPacket[2] << 8) | rtpPacket[3];
}

2 个引用
private int GetRtpTimestamp(byte[] rtpPacket)
{
    // The timestamp in the RTP header is located at 4-7 bytes
    return (rtpPacket[4] << 24) | (rtpPacket[5] << 16) | (rtpPacket[6] << 8) | rtpPacket[7];
}
```

Estos métodos extraen cargas útiles, números de secuencia y marcas de tiempo de los paquetes RTP recibidos para ayudar a ensamblar y procesar los fotogramas de vídeo

Pregunta 5

Indique qué métricas ha implementado. Según su código, comente cómo el cliente realiza el cálculo, detallando qué campos de la cabecera enviados por el servidor son usados (y cómo).

Delay
Jitter
Packet Loss Rat

Cálculo de la latencia y el jitter

En el método VideoThread, el paquete de vídeo recibido extrae la marca de tiempo y calcula la latencia y el jitter del paquete actual

```
private void VideoThread()
{
    while (true)
    {
        try
        {
            byte[] receivedData = videoclient.Receive(ref videoremote);

            // Convert to image
            byte[] imageBytes = GetRtpPayload(receivedData);
            num_video++;

            // Save Serial Number
            int sequenceNumber = GetRtpSequenceNumber(receivedData);
            sequence_video.Add(sequenceNumber);

            // Get timestamp
            long currentTime = DateTimeOffset.Now.ToUnixTimeSeconds();
            int timestamp = GetRtpTimestamp(receivedData);

            // Computational latency and jitter
            if (video_prev_timestamp == 0)
            {
                video_jitter = 0;
            }
            else
            {
                video_delay = (currentTime - video_prev_time) - (timestamp * 0.00001 - video_prev_timestamp * 0.00001);
                video_jitter = video_jitter + (Math.Abs(video_delay) - video_jitter) / 16;
            }

            // Storing information about timestamps and times
            video_prev_timestamp = timestamp;
            video_prev_time = currentTime;

            // Update Tags
            UpdateLabel(label7, $"Delay: {video_delay:0.00} ms");
            UpdateLabel(label8, $"Jitter: {video_jitter:0.00} ms");

            Image frameImage = Image.FromStream(new MemoryStream(imageBytes));
            UpdatePictureBox(pictureBox1, frameImage);

            // Marked as receiving RTP video
            UpdateCheckBox(checkBox2, true);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error receiving video: {ex.Message}");
        }
    }
}
```

Extracción de carga, número de secuencia y marca de tiempo de paquetes RTP

```
2 个引用
private byte[] GetRtpPayload(byte[] rtpPacket)
{
    // Assuming an RTP header length of 12 bytes
    const int headerLength = 12;
    byte[] payload = new byte[rtpPacket.Length - headerLength];
    Array.Copy(rtpPacket, headerLength, payload, 0, payload.Length);
    return payload;
}

2 个引用
private int GetRtpSequenceNumber(byte[] rtpPacket)
{
    // Sequence number in RTP header at 2-3 bytes
    return (rtpPacket[2] << 8) | rtpPacket[3];
}

2 个引用
private int GetRtpTimestamp(byte[] rtpPacket)
{
    // The timestamp in the RTP header is located at 4-7 bytes
    return (rtpPacket[4] << 24) | (rtpPacket[5] << 16) | (rtpPacket[6] << 8) | rtpPacket[7];
}
```

Retardo: se calcula utilizando la diferencia entre la hora actual y la marca de tiempo.

Jitter: el valor del jitter se calcula utilizando el cambio en el retardo

El cálculo de la tasa de pérdida de paquetes se realiza en el método analyzeVideo

```
1 个引用
private void analyzeVideo(object sender, EventArgs e)
{
    total_video += num_video;

    // Calculate Paq/s
    label4.Text = String.Format("{0} paq/s", num_video);
    num_video = 0;

    // Check the sequence numbers
    try
    {
        List<int> copy_video = sequence_video;
        copy_video.Sort();
        List<int> result_video = Enumerable.Range(copy_video.First(), copy_video.Count()).Except(sequence_video).ToList();
        lost_video += result_video.Count();
    }
    catch (Exception)
    {

    }

    sequence_video.Clear();

    // Calculate Paqs lost %
    float video_loss = (lost_video / total_video) * 100;
    label10.Text = String.Format("Loss {0} %", video_loss);
}
```

Metodología de cálculo de los indicadores

Latencia: Se calcula utilizando la hora actual y la marca de tiempo del paquete RTP

```
video_delay = (currentTime - video_prev_time) - (timestamp * 0.00001 - video_prev_timestamp * 0.00001);
video_jitter = video_jitter + (Math.Abs(video_delay) - video_jitter) / 16;
```

Jitter: Utilización del cambio en el retardo para calcular el jitter

Tasa de pérdida de paquetes: Calcula la diferencia entre los paquetes que se espera recibir y los paquetes realmente recibidos. El número de paquetes perdidos se calcula comparando la lista de números de secuencia.

```
// Calculate Paqs lost %
float video_loss = (lost_video / total_video) * 100;
label10.Text = String.Format("Loss {0} %", video_loss);
```

Pregunta 6

Describa, según su código, y en su caso, cómo ha implementado la parte de chat (tanto en el servidor como en el cliente).

Implementación del chat en el servidor

```
private void button1_Click(object sender, EventArgs e)
{
    // Send message to chat
    if (!string.IsNullOrWhiteSpace(richTextBox1.Text))
    {
        // Send to multicast
        try
        {
            string msg = string.Format("{0}: {1}", username, richTextBox1.Text);
            byte[] encodedmsg = Encoding.UTF8.GetBytes(msg);
            chat1server.Send(encodedmsg, encodedmsg.Length, chat1remote);

            // Adding a message to a list box
            listBox1.Items.Add(msg);
            richTextBox1.Clear(); // Clear the contents of the text box
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Error sending the chat message: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

En primer lugar debemos comprobar si richTextBox1 está en blanco, si es así formatearemos el mensaje y lo codificaremos en una matriz de bytes. A continuación, utilizaremos chat1server para enviar el mensaje a chat1remote a través de UDP multicast. por último, añadiremos el mensaje a listBox1 y borraremos el contenido del cuadro de texto

Conectar e inicializar el chat

```
private void button2_Click(object sender, EventArgs e)
{
    // We have clicked the connect button

    // save the chat user
    if (richTextBox2.Text == "")
    {
        richTextBox2.Text = "Anonymous";
        richTextBox2.Enabled = false;
        username = "Anonymous";
    }
    else
    {
        username = richTextBox2.Text;
        richTextBox2.Enabled = false;
    }

    button2.Enabled = false;
    button3.Enabled = true;
    richTextBox1.Enabled = true;
    button1.Enabled = true;
    listBox1.Items.Clear();
    listBox1.Items.Add(String.Format("***  Welcome {0} to the chat room ***", username));

    // Initialising communication channels
    try
    {
        // Video transmission
        videoserver = new UdpClient();
        videoremote = new IPEndPoint(multicast, videoport);
        // Implementing my RTP class
        video = new RTP("video1", videoserver, multicast, videoremote);

        // Timer for paq/s
        timer1 = new Timer();
        timer1.Tick += new EventHandler(analyzeVideo);  //Take parameters from the TX
        timer1.Interval = 1000;                         // Every 1000 ms
        timer1.Start();
```

```
    // Audio transmission
    audioserver = new UdpClient();
    audioremote = new IPEndPoint(multicast, audioport);
    // Implementation my RTP class
    audio = new RTP("audio1", audioserver, multicast, audioremote);

    // We start the audio
    InitAudioCapture();
    audiosender = new Thread(new ThreadStart(SendAudio));

    // Timer for paq/s
    timer2 = new Timer();
    timer2.Tick += new EventHandler(analyzeAudio);  // Take parameters from the TX
    timer2.Interval = 1000;                         //  Every 1000 ms
    timer2.Start();

    // Send chat
    chat1server = new UdpClient();
    chat1remote = new IPEndPoint(multicast, chat1port);
    chat1server.JoinMulticastGroup(multicast);

    // Receive chat
    chat2server = new UdpClient(chat2port);
    chat2remote = null;
    chat2server.JoinMulticastGroup(multicast);

    checkBox4.Checked = true;

    Thread t = new Thread(this.ChatThread);
    t.Start();
    CheckForIllegalCrossThreadCalls = false;
}
catch (Exception excp)
{
    MessageBox.Show(excp.ToString());
```

En primer lugar debe guardarse el nombre de usuario. Si richTextBox2 está vacío, configúralo como "Anónimo" y luego activa/desactiva los controles relacionados.

Inicializa el cliente UDP y el grupo de multidifusión para iniciar un nuevo hilo para recibir mensajes de chat.

Recibir mensajes de chat

```
1 个引用
private void ChatThread()
{
    // keep the socket on standby
    while (true)
    {
        byte[] received = chat2server.Receive(ref chat2remote);

        msg = Encoding.UTF8.GetString(received, 0, received.Length);
        listBox1.Items.Add(msg);
    }
}
```

Introduzca un bucle infinito para recibir mensajes continuamente. Utiliza chat2server para recibir mensajes UDP, decodifica los mensajes recibidos en cadenas y los añade a listBox1

Implementación del Chat en el Lado Cliente

Envío de mensajes de chat

```
1 个引用
private void button1_Click(object sender, EventArgs e)
{
    // Send message to chat
    if (!string.IsNullOrWhiteSpace(richTextBox1.Text))
    {
        // Send to multicast
        try
        {
            string message = $"{username}: {richTextBox1.Text}";
            byte[] encodedMessage = Encoding.UTF8.GetBytes(message);
            chat2client.Send(encodedMessage, encodedMessage.Length, chat2remote);

            // Adding a message to a list box
            listBox1.Items.Add(message);
            richTextBox1.Clear();
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Error sending the chat message: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

Conectar e inicializar el chat

```
private void button3_Click(object sender, EventArgs e)
{
    // Click the Connect button

    // Save Chat User
    if (string.IsNullOrWhiteSpace(richTextBox2.Text))
    {
        richTextBox2.Text = "AnonymousClient";
        username = "AnonymousClient";
    }
    else
    {
        username = richTextBox2.Text;
    }
    richTextBox2.Enabled = false;

    // Update button and control status
    button2.Enabled = true;
    button3.Enabled = false;
    richTextBox1.Enabled = true;
    button1.Enabled = true;
    listBox1.Items.Clear();
    listBox1.Items.Add($"*** Welcome {username} to the chat room ***");

    // Initialising the communication channel
    try
    {
        InitializeVideoReception();
        InitializeAudioReception();
        InitializeChatReception();
        InitializeChatSending();

        // Start chat thread
        Thread chatThread = new Thread(ChatThread);
        chatThread.Start();
        CheckForIllegalCrossThreadCalls = false;
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Error initializing communication channels: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        checkBox4.Checked = true;
    }
}
```

Guarda el nombre de usuario. Si richTextBox2 está vacío, configúrelo como "AnonymousClient"

Inicializar el cliente UDP y el grupo de multidifusión para recibir y enviar mensajes de chat. Iniciar un nuevo hilo para recibir mensajes de chat

Recibir mensajes de chat

```
1 个引用
private void ChatThread()
{
    // We keep the socket on standby
    while (true)
    {
        byte[] received = chat1client.Receive(ref chat1remote);

        msg = Encoding.UTF8.GetString(received);
        listBox1.Items.Add(msg);
    }
}
```

Pregunta 7

Describa, según su código, y en su caso, cómo ha implementado la parte de transmisión/recepción del flujo de audio.

Implementación de streaming de audio en el servidor

Inicialización de la captura de audio

```
1 个引用
private void InitAudioCapture()
{
    // Set up DirectSound
    device = new Device();
    device.SetCooperativeLevel(this, CooperativeLevel.Normal);

    capture = new Capture(record_source);

    // Create Waveformat
    waveFormat = new WaveFormat
    {
        BitsPerSample = bitsPerSample,                                          // 16 bits
        BlockAlign = (short)(channels * (bitsPerSample / (short)8)),
        Channels = channels,                                                    // Stereo
        AverageBytesPerSecond = (short)(channels * (bitsPerSample / (short)8)) * samplesPerSecond,      // 22kHz
        SamplesPerSecond = samplesPerSecond,                                    // 22kHz
        FormatTag = WaveFormatTag.Pcm
    };
    // Create CaptureBufferDescription
    captureBuffDesc = new CaptureBufferDescription
    {
        BufferBytes = waveFormat.AverageBytesPerSecond / 5,
        Format = waveFormat
    };

    bufferDesc = new BufferDescription
    {
        BufferBytes = waveFormat.AverageBytesPerSecond / 5,
        Format = waveFormat
    };

    bufferplayback = new SecondaryBuffer(bufferDesc, device);
    buffersize = captureBuffDesc.BufferBytes;
    }
```

Primero configure el dispositivo DirectSound y configure el nivel de cooperación. A continuación, inicialice el dispositivo de captura de audio y el formato de forma de onda. Por último, crea la descripción del búfer de captura y la descripción del búfer de reproducción

Creación de un lugar de notificación

```
1 个引用
private void CreateNotifyPositions()
{
    try
    {
        autoResetEvent = new AutoResetEvent(false);
        notify = new Notify(captureBuffer);
        BufferPositionNotify bufferPositionNotify1 = new BufferPositionNotify
        {
            Offset = buffersize / 2 - 1,
            EventNotifyHandle = autoResetEvent.SafeWaitHandle.DangerousGetHandle()
        };
        BufferPositionNotify bufferPositionNotify2 = new BufferPositionNotify
        {
            Offset = buffersize - 1,
            EventNotifyHandle = autoResetEvent.SafeWaitHandle.DangerousGetHandle()
        };

        notify.SetNotificationPositions(new BufferPositionNotify[] { bufferPositionNotify1, bufferPositionNotify2 });
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Error on CreatePositionNotify", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Crear eventos de auto-reset para notificación

Establecer notificación de posición de búfer para captura de datos de audio

SendAudio

```
1 个引用
private void SendAudio()
{
    try
    {
        // We capture the audio and send it over the network.
        int halfbuffer = buffersize / 2;
        captureBuffer = new CaptureBuffer(captureBuffDesc, capture);
        CreateNotifyPositions();
        captureBuffer.Start(true);
        bool readFirstBufferPart = true;
        int offset = 0;
        MemoryStream memStream = new MemoryStream(halfbuffer);

        while (!button7.Enabled)
        {
            // We look forward to an event
            autoResetEvent.WaitOne();
            // We put the pointer at the beginning of the MS
            memStream.Seek(0, SeekOrigin.Begin);
            // We read the Capture Buffer and save it in the first half.
            captureBuffer.Read(offset, memStream, halfbuffer, LockFlag.None);
            readFirstBufferPart = !readFirstBufferPart;
            offset = readFirstBufferPart ? 0 : halfbuffer;

            // Prepare the data stream
            //byte[] data = memStream.GetBuffer(); // No compression
            byte[] data = ALawEncoder.ALawEncode(memStream.GetBuffer());

            // We send via RTP to the user.
            audio.sendALaw(data);
            num_audio++;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Error sending audio: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
```

Aquí se incluyen varios aspectos, en primer lugar la inicialización del buffer de captura y la creación de la ubicación de notificación. Luego comienza a capturar los datos de audio y lee los datos en el flujo de memoria.

Comprimir los datos de audio utilizando codificación A-Law. Finalmente envía los datos de audio comprimidos vía RTP

Implementación de la recepción de flujos de audio en el lado del cliente

Inicialización de la recepción de audio

```
1 个引用
private void InitializeAudioReception()
{
    // audio reception
    audioclient = new UdpClient();
    audioremote = new IPEndPoint(IPAddress.Any, audioport);
    audioclient.Client.Bind(audioremote);
    audioclient.JoinMulticastGroup(multicast);

    // Initialising the audio
    InitCaptureSound();
    lost_audio = 0;
    sequence_audio = new List<int>();

    receivedAudio = new Thread(AudioThread);
    receivedAudio.Start();

    // timer
    timer2 = new Timer
    {
        Interval = 1000 // Every 1000 ms
    };
    timer2.Tick += analyzeAudio;
    timer2.Start();

    // Audio QoA initialisation
    audio_prev_time = 0;
    audio_prev_timestamp = 0;
}
```

Inicializar clientes UDP y unirse a grupos multicast.

Inicializar la configuración de captura de audio.

Iniciar el hilo de recepción de audio.

Iniciar temporizador para analizar la calidad de audio

Inicialización de la captura de audio

```
1 个引用
public void InitCaptureSound()
{
    device = new Device();
    device.SetCooperativeLevel(this, CooperativeLevel.Normal);

    // We create the WaveFormat
    waveFormat = new WaveFormat
    {
        BitsPerSample = bitsPerSample,                                                  // 16 bits
        BlockAlign = (short)(channels * (bitsPerSample / (short)8)),
        Channels = channels,                                                            // Stereo
        AverageBytesPerSecond = (short)(channels * (bitsPerSample / (short)8)) * samplesPerSecond,  // 22kHz
        SamplesPerSecond = samplesPerSecond,                                            // 22kHz
        FormatTag = WaveFormatTag.Pcm
    };

    bufferDesc = new BufferDescription
    {
        BufferBytes = waveFormat.AverageBytesPerSecond / 5,
        Format = waveFormat
    };

    bufferplayback = new SecondaryBuffer(bufferDesc, device);
}
```

Configurar el dispositivo DirectSound y configurar el nivel de cooperación.

Inicializar el formato de onda de audio.

Crear la descripción del búfer de reproducción e inicializar el búfer de reproducción

Audio Receiver Thread

```
{
    try
    {
        receivedData = audioclient.Receive(ref audioremote);

        byte[] audioBytes = GetRtpPayload(receivedData);
        num_audio++;

        int sequenceNumber = GetRtpSequenceNumber(receivedData);
        sequence_audio.Add(sequenceNumber);

        long currentTime = DateTimeOffset.Now.ToUnixTimeSeconds();
        int timestamp = GetRtpTimestamp(receivedData);

        if (audio_prev_timestamp == 0)
        {
            audio_jitter = 0;
        }
        else
        {
            audio_delay = (currentTime - audio_prev_time) - (timestamp * 0.000125 - audio_prev_timestamp * 0.000125);
            audio_jitter = audio_jitter + (Math.Abs(audio_delay) - audio_jitter) / 16;
        }
        audio_prev_timestamp = timestamp;
        audio_prev_time = currentTime;

        // Update Tags
        UpdateLabel(label5, $"Delay: {audio_delay:0.00} ms");
        UpdateLabel(label6, $"Jitter: {audio_jitter:0.00} ms");

        // Marked as receiving RTP audio
        UpdateCheckBox(checkBox1, true);

        // Decode audio data
        byte[] decodedAudioData = new byte[audioBytes.Length * 2];
        ALawDecoder.ALawDecode(audioBytes, out decodedAudioData);

        // Playing audio data
        UpdateCheckBox(checkBox3, true);
        bufferplayback = new SecondaryBuffer(bufferDesc, device);
        bufferplayback.Write(0, decodedAudioData, LockFlag.None);
        bufferplayback.Play(0, BufferPlayFlags.Default);
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error receiving audio: {ex.Message}");
        return;
```

En primer lugar debemos entrar en un bucle infinito para recibir continuamente datos de

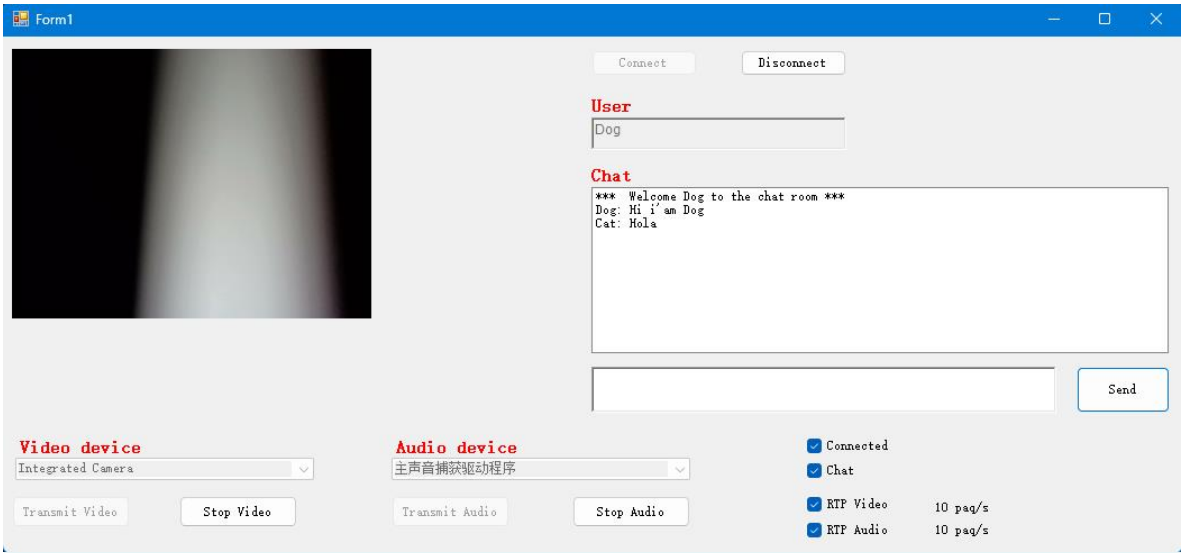
audio. A partir de ahí, extraer la carga útil, el número de secuencia y la marca de tiempo del paquete RTP. A continuación, calcular el retardo y el jitter del audio.

Decodifica los datos de audio utilizando A-Law. Escribe los datos de audio decodificados en un búfer y reprodúcelos.

Pregunta 8

Suba dos capturas de pantalla de su aplicación servidora y cliente en funcionamiento (en este caso es obligatorio subir ficheros en formato jpeg, gif o png).

Server



Client